#3
5-14-02

Docket No.: 57454-180 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	
Toyohiko YOSHIDA	:	
Serial No.:	:	Group Art Unit:
Filed: July 25, 2001	:	Examiner:

For: DATA PROCESSING DEVICE WITH INSTRUCTION TRANSLATOR AND MEMORY INTERFACE DEVICE

CLAIM OF PRIORITY AND TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Commissioner for Patents
Washington, DC 20231

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicant hereby claims the priority of:

Japanese Patent Application No. 2001-012687, filed January 22, 2001

cited in the Declaration of the present application. A Certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Stephen A. Becker
Registration No. 26,527

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 SAB:prp
Date: July 25, 2001
Facsimile: (202) 756-8087

5145T-18U
YOSHIDA
July 25, 2001

日 本 国 特 許 庁

McDermott, Will & Emery

PATENT OFFICE
JAPANESE GOVERNMENT

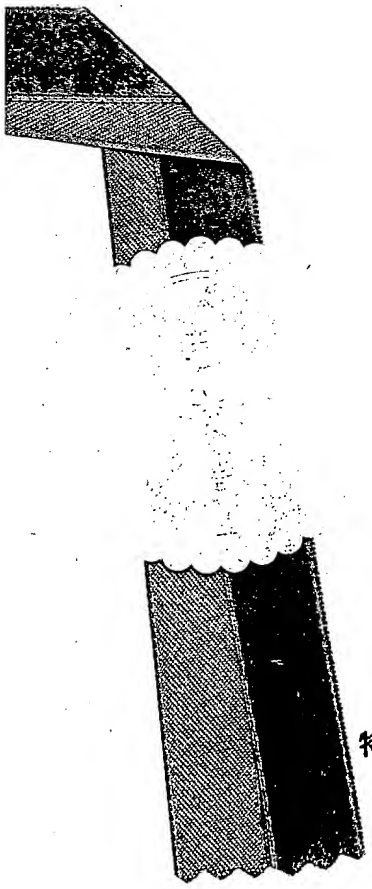
別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２００１年 １月２２日

出 願 番 号
Application Number: 特願２００１－０１２６８７

出 願 人
Applicant (s): 三菱電機株式会社



CERTIFIED COPY OF
PRIORITY DOCUMENT

２００１年 ２月 ９日

特許庁長官
Commissioner,
Patent Office

及川耕造

出証番号 出証特２００１－３００６７５５

【書類名】　特許願
【整理番号】　527420JP01
【提出日】　平成13年 1月22日
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/30
　　G06F　9/455
【発明者】
　【住所又は居所】　東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内
　【氏名】　吉田　豊彦
【特許出願人】
　【識別番号】　000006013
　【氏名又は名称】　三菱電機株式会社
【代理人】
　【識別番号】　100064746
　【弁理士】
　【氏名又は名称】　深見　久郎
【選任した代理人】
　【識別番号】　100085132
　【弁理士】
　【氏名又は名称】　森田　俊雄
【選任した代理人】
　【識別番号】　100091409
　【弁理士】
　【氏名又は名称】　伊藤　英彦
【選任した代理人】
　【識別番号】　100096781
　【弁理士】
　【氏名又は名称】　堀井　豊

【選任した代理人】

【識別番号】　100096792

【弁理士】

【氏名又は名称】　森下　八郎

【手数料の表示】

【予納台帳番号】　008693

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　命令トランスレータを備えたデータ処理装置およびメモリインタフェース装置

【特許請求の範囲】

【請求項１】　プロセッサコアと、

前記プロセッサコアと、所定の外部メモリ空間にマップされる外部メモリとの間に設けられるメモリインタフェース部とを含み、

前記メモリインタフェース部は、

前記プロセッサコアから前記外部メモリ空間へのアクセスのためのアドレス値を受け、前記外部メモリから命令をフェッチするための命令フェッチ手段と、

前記外部メモリから受ける、前記プロセッサコアに対する非ネイティブ命令をネイティブ命令に変換するための変換器と、

前記プロセッサコアから前記外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、前記外部メモリ空間から読出された命令と、前記外部メモリ空間から読み出された命令を前記変換器で変換した結果の命令とのいずれかを選択的に前記プロセッサコアに与えるための選択手段とを含む、命令トランスレータを備えたデータ処理装置。

【請求項２】　前記命令フェッチ手段は、

前記プロセッサコアから前記外部メモリ空間へのアクセス時のアドレスに対して所定の変換を行なうためのアドレス変換手段と、

前記アドレスが、前記所定領域内にあるか否かにしたがって、前記プロセッサコアからのアドレスと、前記アドレス変換手段の出力するアドレスとのいずれかを選択的に前記外部メモリに与えるための手段とを含む、請求項１に記載の命令トランスレータを備えたデータ処理装置。

【請求項３】　前記変換手段は、入力されたアドレス値を２のｎ乗（ｎは自然数）で除して出力する除算手段を含む、請求項２に記載の命令トランスレータを備えたデータ処理装置。

【請求項４】　前記除算手段は、入力されたアドレス値をｎビット右シフトするシフタを含む、請求項３に記載の命令トランスレータを備えたデータ処理装

置。

【請求項５】　プロセッサコアと、所定の外部メモリ空間にマップされる外部メモリとの間に設けられるメモリインタフェース装置であって、

前記プロセッサコアから前記外部メモリ空間へのアクセスのためのアドレス値を受け、前記外部メモリから命令をフェッチするための命令フェッチ手段と、

前記外部メモリから受ける、前記プロセッサコアに対する非ネイティブ命令をネイティブ命令に変換するための変換器と、

前記プロセッサコアから前記外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、前記外部メモリ空間から読出された命令と、前記外部メモリ空間から読み出された命令を前記変換器で変換した結果の命令とのいずれかを選択的に前記プロセッサコアに与えるための選択手段とを含む、命令トランスレータを備えたメモリインタフェース装置。

【請求項６】　前記命令フェッチ手段は、

前記プロセッサコアから前記外部メモリ空間へのアクセス時のアドレスに対して所定の変換を行なうためのアドレス変換手段と、

前記アドレスが、前記所定領域内にあるか否かにしたがって、前記プロセッサコアからのアドレスと、前記アドレス変換手段の出力するアドレスとのいずれかを選択的に前記外部メモリに与えるための手段とを含む、請求項５に記載の命令トランスレータを備えたメモリインタフェース装置。

【請求項７】　前記変換手段は、入力されたアドレス値を２のｎ乗（ｎは自然数）で除して出力する除算手段を含む、請求項６に記載の命令トランスレータを備えたメモリインタフェース装置。

【請求項８】　前記除算手段は、入力されたアドレス値をｎビット右シフトするシフタを含む、請求項７に記載の命令トランスレータを備えたメモリインタフェース装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明はあるプロセッサにとっての非ネイティブ命令を、そのプロセッサの

ネイティブ命令に変換するための命令トランスレータを備え、非ネイティブ命令からなる処理ルーチンとネイティブ命令からなる処理ルーチンの混在したプログラムを高速に実行することが可能なデータ処理装置とそのためのメモリインタフェース装置とに関する。

【０００２】

【従来の技術】

プロセッサアーキテクチャとそのプロセッサで実行可能な命令体系とは密接な関係を持つ。一方、プロセッサアーキテクチャが進化して命令体系が新しくなると、旧命令体系で実現されたプログラムコードはそのままでは実行不可能となることが通常である。どのようにして旧命令体系で実現されたプログラム資産を有効に承継するかが問題となる。そのため、ある命令体系を持つ新プロセッサで、旧命令体系にしたがって設計された旧プロセッサ用に記述されたプログラムを実行するようにするための手法が多く開発されている。

【０００３】

旧プロセッサ用に記述されたプログラムを新プロセッサで実行するために従来行なわれている代表的な方法に、新プロセッサのハードウェアに旧プロセッサの機能を持たせる方法がある。図２１を参照して、そうした方法を実現する従来のデータ処理装置５００は、旧プロセッサの命令と新プロセッサの命令との双方をデコードする機能を有する多機能命令デコーダ５およびそれら命令を実行する機能を有する演算部６を備えたプロセッサ１と、プロセッサ１に接続されるバス４と、バス４に接続されるデータメモリ２および命令メモリ３とを含む。

【０００４】

命令メモリ３は、旧プロセッサの命令と新プロセッサの命令との双方を保持する。多機能命令デコーダ５は、命令メモリ３から読出され、バス４を介してプロセッサ１に転送された命令をデコードする。このとき、多機能命令デコーダ５は、この命令が新プロセッサ用のものであっても、旧プロセッサ用のものであってもデコードする機能を有する。デコードされた命令は演算部６によって実行される。データメモリ２には、新プロセッサ用の命令からも、旧プロセッサ用の命令からもアクセスできる。

【０００５】

このように新プロセッサのハードウェアに旧プロセッサのハードウェアの機能を持たせる例は、たとえば「IA-32 Application Execution Model in an IA-64 System Environment」（IA-64 Application Developer & Architecture Guide, Chapter 6, May 1999）に詳しく記載されている。

【０００６】

旧プロセッサ用に記述されたプログラムを新プロセッサで実行するための従来の他の方法として、旧プロセッサ用のソフトウェアを新プロセッサ用のソフトウェアに変換した後に実行する方法、および旧プロセッサ用の命令の動作を新プロセッサ用のソフトウェアでエミュレートする方法がある。この方法は例えば、Tom Thompsonによる「An Alpha in PC Clothing (Digital Equipment's new x86 emulator technology makes an Alpha system a fast x86 clone)」（BYTE, pp. 195-196, February 1996）に詳しく記載されている。

【０００７】

ところで、ある命令体系で記述されたプログラムを他の命令体系にしたがって設計されたプロセッサで実行することは、これ以外の場合にも有効である。たとえば、ある命令体系のサブセットを定義してその縮小された命令体系でプログラムを記述すると、プログラムサイズを小さくすることができる。また、ＪＡＶＡ™（Java及びすべてのJava関連の商標及びロゴは、米国及びその他の国における米国Sun Microsystems, Incの商標又は登録商標である。）言語は、仮想的なプロセッサの命令体系を定めてプログラムを記述し、その同一のプログラムを複数のプロセッサでそれぞれのプロセッサの命令体系を用いて実行する仕組みとなっている。このためＪＡＶＡ言語で記述されたプログラムは、異なる命令体系の複数種類のプロセッサで共有して実行することができる。

【０００８】

プログラムサイズを小さくする目的でサブセットの縮小命令体系を作り、プロセッサの多機能命令デコーダで非縮小命令体系の命令と縮小命令体系の命令との双方の命令をデコードする方法は既に多数提案されている。たとえばJames L. Turleyによる「Thumb Squeezes ARM Code Size (New Core Module Provides Opti

mized Second Instruction Set)」（Micro Processor Report, Vol. 9, No. 4, pp. 1, 6-9, March 27, 1995）の記載を参照。

【０００９】

【発明が解決しようとする課題】

しかし、上述したような従来の手法には、いずれも以下に述べるような問題点がある。

【００１０】

複数個の命令体系によって記述されたプログラムを実行する機能をプロセッサのハードウェアに持たせる場合、ハードウェアが複雑になり、かつそのサイズが大きくなる。また、実行すべき命令体系を追加したり変更したりする場合には、ハードウェア全体を設計し直す必要があり、柔軟に対応することが困難である。

【００１１】

ソフトウェアによりプログラムを変換する場合には次のような問題がある。プログラム自体を変換する場合には、変換後プログラムを保持するために大容量のメモリを新たに必要とする。その結果、メモリのコストが増大しデータ処理装置のコストも上昇する。また命令の動作を他の命令体系の命令でエミュレートする場合には、演算結果をエミュレートすることが必要であることはもちろん、プログラムカウンタの値、および必要がある場合にはフラグまでエミュレートしなければならない。そのため、一つの命令の動作を、別の体系の多数の命令で置換することが必要になる。その結果、動作速度が大幅に低下するという問題がある。

【００１２】

この発明はこうした問題を解決するためになされたもので、プロセッサのハードウェア自体は変更せずに、複数個の異なる命令体系の命令からなるプログラムをネイティブ命令を用いて高速に実行可能なデータ処理装置であって、大容量のメモリを必要としないデータ処理装置、およびそのための命令トランスレータを備えたメモリインターフェース装置を提供することを目的とする。

【００１３】

【課題を解決するための手段】

請求項１に記載の発明にかかるデータ処理装置は、プロセッサコアと、プロセ

ッサコアと所定の外部メモリ空間にマップされる外部メモリとの間に設けられるメモリインタフェース部とを含み、メモリインタフェース部は、プロセッサコアから外部メモリ空間へのアクセスのためのアドレス値を受け、外部メモリから命令をフェッチするための命令フェッチ手段と、外部メモリから受ける、プロセッサコアに対する非ネイティブ命令をネイティブ命令に変換するための変換器と、プロセッサコアから外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、外部メモリ空間から読出された命令と、外部メモリ空間から読み出された命令を変換器で変換した結果の命令とのいずれかを選択的にプロセッサコアに与えるための選択手段とを含む。

【００１４】

プロセッサ外部では非ネイティブ命令およびネイティブ命令がバス上を転送されるが、メモリインタフェース部において非ネイティブ命令はネイティブ命令に変換される。そのためプロセッサコア内部ではネイティブ命令のみが転送される。プロセッサコアは実行中の命令がネイティブ命令か、非ネイティブ命令かを判断することなく命令を単純に実行することにより、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを実行することができる。プロセッサコア内部ではネイティブ命令のみなので、実行も高速である。

【００１５】

請求項２に記載の発明にかかるデータ処理装置は、請求項１に記載の発明の構成に加えて、命令フェッチ手段は、プロセッサコアから外部メモリ空間へのアクセス時のアドレスに対して所定の変換を行なうためのアドレス変換手段と、アドレスが、所定領域内にあるか否かにしたがって、プロセッサコアからのアドレスと、アドレス変換手段の出力するアドレスとのいずれかを選択的に外部メモリに与えるための手段とを含む。

【００１６】

非ネイティブ命令の命令長とネイティブ命令の命令長とが異なる場合でも、プロセッサコアは命令長の違いを意識する必要がない。

【００１７】

請求項３に記載の発明にかかるデータ処理装置は、請求項２に記載の発明の構

成に加えて、変換手段は、入力されたアドレス値を２のｎ乗（ｎは自然数）で除して出力する除算手段を含む。

【００１８】

非ネイティブ命令の命令長がネイティブ命令の命令長を２のｎ乗で除した大きさのとき、単純な回路だけで非ネイティブ命令をフェッチするためのアドレス変換を行なうことができる。

【００１９】

請求項４に記載の発明にかかるデータ処理装置は、請求項３に記載の発明の構成に加えて、除算手段は、入力されたアドレス値をｎビット右シフトするシフタを含む。

【００２０】

非常に単純な回路で非ネイティブ命令をフェッチするためのアドレス変換が行なえる。

【００２１】

請求項５に記載の発明にかかるメモリインタフェース装置は、プロセッサコアと、所定の外部メモリ空間にマップされる外部メモリとの間に設けられるメモリインタフェース装置であって、プロセッサコアから外部メモリ空間へのアクセスのためのアドレス値を受け、外部メモリから命令をフェッチするための命令フェッチ手段と、外部メモリから受ける、プロセッサコアに対する非ネイティブ命令をネイティブ命令に変換するための変換器と、プロセッサコアから外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、外部メモリ空間から読出された命令と、外部メモリ空間から読み出された命令を変換器で変換した結果の命令とのいずれかを選択的にプロセッサコアに与えるための選択手段とを含む。

【００２２】

プロセッサ外部では非ネイティブ命令およびネイティブ命令がバス上を転送されるが、メモリインタフェース部において非ネイティブ命令はネイティブ命令に変換される。そのためプロセッサコア内部ではネイティブ命令のみが転送される。プロセッサコアは実行中の命令がネイティブ命令か、非ネイティブ命令かを判

断することなく命令を単純に実行することにより、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを実行することができる。プロセッサコア内部ではネイティブ命令のみなので、プロセッサコアにおける実行を高速にすることができる。

【００２３】

請求項６に記載の発明にかかるメモリインタフェース装置は、請求項５に記載の発明の構成に加えて、命令フェッチ手段は、プロセッサコアから外部メモリ空間へのアクセス時のアドレスに対して所定の変換を行なうためのアドレス変換手段と、アドレスが、所定領域内にあるか否かにしたがって、プロセッサコアからのアドレスと、アドレス変換手段の出力するアドレスとのいずれかを選択的に外部メモリに与えるための手段とを含む。

【００２４】

非ネイティブ命令の命令長とネイティブ命令の命令長とが異なる場合でも、プロセッサコアは命令長の違いを意識する必要がない。

【００２５】

請求項７に記載の発明にかかるメモリインタフェース装置は、請求項６に記載の発明の構成に加えて、変換手段は、入力されたアドレス値を２のｎ乗（ｎは自然数）で除して出力する除算手段を含む。

【００２６】

非ネイティブ命令の命令長がネイティブ命令の命令長を２のｎ乗で除した大きさのとき、単純な回路で非ネイティブ命令をフェッチするためのアドレス変換を行なうことができる。

【００２７】

請求項８に記載の発明にかかるメモリインタフェース装置は、請求項７に記載の発明の構成に加えて、除算手段は、入力されたアドレス値をｎビット右シフトするシフタを含む。

【００２８】

非常に単純な回路で非ネイティブ命令をフェッチするためのアドレス変換が行なえる。

【００２９】

【発明の実施の形態】

図１を参照して、この発明の実施の形態のデータ処理装置は、半導体集積回路上に形成されたプロセッサ５１０と、プロセッサ５１０に接続されたアドレスバス３１、データバス３２および制御バス３３と、いずれもアドレスバス３１、データバス３２および制御バス３３に接続された、データメモリ５２１、ネイティブ命令用メモリ５２２、ＪＡＶＡ命令メモリ５２３、およびＲＡＭ（ランダムアクセスメモリ）５２０とを含む。

【００３０】

図２を参照して、プロセッサ５１０は、半導体集積回路の１チップからなり、プロセッサのコア１００と、命令メモリ５０１と、データメモリ５０２と、アドレスバス３１、データバス３２および制御バス３３に接続されたバスインターフェイス部５０３と、これらを互いに接続してアドレスおよびデータを送信するためのデータアドレスバス１０６およびデータバス１０７と、コア１００、命令メモリ５０１およびバスインターフェイス部５０３を相互に接続する６４ビット幅の命令アドレスバス１０４および６４ビット幅の命令バス１０５とを含む。

【００３１】

コア１００は、ＶＬＩＷ（Very Long Instruction Word）型命令体系を有するプロセッサコアである。コア１００は、命令バス１０５から入力されたＶＬＩＷ命令をデコードするための命令デコーダ１１０と、命令デコーダ１１０によってデコードされた命令を実行するためのメモリ演算部１３０および整数演算部１４０と、メモリ演算部１３０および整数演算部１４０に複数のバスで接続された、ソフトウェアで読み書きできる汎用レジスタファイル１２０とを含む。

【００３２】

命令デコーダ１１０は二つのサブ命令デコーダ１１１および１１２を含む。

メモリ演算部１３０は、アドレス演算器１３１、ＰＣ（プログラムカウンタ）演算器１３２、シフタ１３３およびＡＬＵ（Arithmetic Logic Unit）１３４などの演算器を含む。メモリ演算部１３０は、サブ命令デコーダ１１１の出力にしたがい、メモリアクセス命令、ＰＣ制御命令、整数演算命令などを実行するため

のものである。整数演算部１４０は、シフタ１４１、ＡＬＵ１４２、乗算器１４３およびアキュムレータ１４４を含む。整数演算部１４０は、サブ命令デコーダ１１２の出力にしたがい整数演算命令を実行するためのものである。メモリ演算部１３０および整数演算部１４０は、２つのサブ命令を並列に実行する場合と、それぞれ独立に１つのサブ命令を実行する場合とがある。

【００３３】

図３を参照して、プロセッサ５１０が有するレジスタファイル１２０は、６４本の３２ビット汎用レジスタであるレジスタ１５０～１５２、…、１６２、１６３を含む。なお後述するようにレジスタ１６３は二通りに使い分けられるので、それらをそれぞれレジスタ１６３ａおよび１６３ｂとして便宜上別々のものとして示してある。プロセッサ５１０はさらに、制御レジスタ１７０～１８０を含む。また、図２に示すアキュムレータ１４４はそれぞれ６４ビットの二つのアキュムレータ１４４ａおよびアキュムレータ１４４ｂを含む。

【００３４】

レジスタ１５０は常にゼロを保持するレジスタである。レジスタ１６２は非割込処理中のスタックトップのデータを保持するためのものである。レジスタ１６３ｂは非割込処理中のスタックポインタでスタックトップのすぐ下のデータのアドレスを保持するためのものである。

【００３５】

レジスタ１６３は、ＰＳＷ（プロセッサステータスワード）である制御レジスタ１７０中にあるモードビットで切り替わり、割込処理中はレジスタ１６３ａとして使用され、非割込処理中はレジスタ１６３ｂとして使用される。

【００３６】

制御レジスタ１７０～１８０は、それぞれ所定の要素のための専用のレジスタである。たとえば制御レジスタ１７０はＰＳＷであって、演算により変化するフラグ、割込処理処理中か否か、割込マスク中か否か、デバック中か否かなど、プロセッサ５１０の動作モードを示すモードビットを含む。制御レジスタ１７２はプログラムカウンタ（ＰＣ）であり、現在実行中の命令のアドレスを示す。制御レジスタ１７１および１７３は、割込受付時、例外発生時、トラップ発生時にそ

れぞれ制御レジスタ１７０および１７２の値をコピーして保持するためのものである。そのほかの制御レジスタの役割に付いては図３に記載したとおりである。それらは本発明と大きな関連を持たないので、記載を簡単にする目的でここではそれらについての詳細な説明は行なわない。

【００３７】

アキュムレータ１４４ａおよび１４４ｂは、乗算結果、積和演算結果を保持するためのものである。アキュムレータ１４４ａおよび１４４ｂはそれぞれ、汎用レジスタの２倍のビット長である６４ビットのデータを保持することができる。

【００３８】

図４を参照して、制御レジスタ１７０が保持するＰＳＷは３２ビットであって、割込処理中か非割込処理中かを示すモードビットであるＳＭビット１７０ａと、割込許可中か割込禁止中かを示すＩＥビット１７０ｂと、命令の実行条件を制御するＦ０ビット１７０ｃおよびＦ１ビット１７０ｄとを含む。この他に制御レジスタ１７０は、ＲＰビット、ＭＤビット、およびＦ２～Ｆ７の各ビットを含む。これらの意味については図４に示す通りである。割込み許可中はプロセッサ１５０へ外部から割込みが要求されているとき、ＶＬＩＷ命令の切れ目で割込みが受け付けられる。

【００３９】

図５を参照して、プロセッサ５１０のバスインターフェース部５０３は、ＪＡＶＡバイトコードをネイティブ命令にハードウェア（以下単に「ＨＷ」と呼ぶ。）トランスレート、すなわちハードウェア回路により変換するためのトランスレーション回路５４０と、制御バス３３に接続され、ＨＷトランスレーションの制御およびプロセッサ５１０のメモリ５２０～５２３へのアクセスを制御するための、アドレスレジスタ５３３を有する制御部５３２と、命令アドレスバス１０４およびデータアドレスバス１０６にそれぞれ接続された二つの入力と、信号線５４２に接続された出力とを有するセレクタ５３１と、信号線５４２上のアドレスを８で除して、すなわち３ビット右シフトして、信号線５４３上に出力するためのアドレス信号シフト回路５３４と、信号線５４３および５４２にそれぞれ接続された二つの入力と、信号線５４５に接続された出力とを有し、トランスレーシ

ョン回路５４０および制御部５３２から信号線５４６を介して与えられる信号により制御されるセレクタ５３５とを含む。

【００４０】

バスインターフェース部５０３はさらに、トランスレーション回路の出力に接続された信号線５４４と、信号線５４５とにそれぞれ接続された二つの入力と、アドレスバス３１の下位２４ビットに接続された出力とを有するセレクタ５３８と、信号線５６２と４バイト幅の信号線５４１とにそれぞれ接続された選択側の２端子と、４バイト幅のデータバス３２に接続された１端子とを有し、データバス３２からの入力を信号線５４１または５６２のいずれかに選択的に出力し、また信号線５６２上のデータを選択してデータバス３２上に出力する機能を有するセレクタ５３７と、信号線５４１に接続された入力と信号線５６０に接続された出力とを有するバス幅変換回路５６０と、トランスレーション回路５４０からの出力線５３９およびバス幅変換回路５６０からの８バイト幅の信号線５６３にそれぞれ接続された二つの入力と、命令バス１０５に接続された出力とを有するセレクタ５３６と、４バイト幅の信号線５６２と６４ビット幅のデータバス１０７とにそれぞれ接続された二つの入出力端子を有するバス幅変換回路５６１とを含む。

【００４１】

信号線５４１はトランスレーション回路５４０の入力にも接続されている。信号線５４２は、制御部５３２にも接続されている。

【００４２】

制御部５３２は、信号線５３１から受けるアドレス値が所定のアドレス領域内にあるかどうかを判断し、この判断に応じてセレクタ５３５、５３８および５３６を制御する。ここで所定のアドレス領域とは後述する図１２のアドレス領域５３７に相当する。制御部５３２は受取るアドレス値がその所定のアドレス領域内にあるとき、信号線５４３を選択して信号線５４３のアドレス値（シフト回路５３４により信号線５４２のアドレス値が変換されたもの）を信号線５４５に与え、そうでないときには、信号線５４２を選択して信号線５４２のアドレス値を信号線５４５に与えるようにセレクタ５３５を制御する。また制御部５３２は、受

取るアドレス値がその所定のアドレス領域内にあるとき、信号線５３９を選択して信号線５３９の命令（トランスレーション回路５３９により非ネイティブ命令が変換された後のネイティブ命令）を信号線１０４に与え、そうでないときには信号線５６３を選択して信号線５４１の命令（ネイティブ命令）を信号線１０４に与えるようにセレクタ５３６を制御する。また制御部５３２は、受取るアドレス値がその所定のアドレス領域内にあるとき、信号線５４４を選択して信号線５４４のアドレス値を信号線３１に与え、そうでないときには信号線５４５を選択して信号線５４５のアドレス値を信号線３１に与えるようにセレクタ５３８を制御する。

【００４３】

またプロセッサコア１００は、メモリをアクセスするとき、命令アドレスで指定される命令に対するものか、またはデータアドレスで指定されるデータに対するものかを示す制御信号を出力する。セレクタ５３１は、この制御信号によって制御され、命令に対するアクセスのときは信号線１０４を選択し信号線１０４のアドレス値を信号線５４２に与え、データに対するアクセスのときは信号線１０６を選択し信号線１０６のアドレス値を信号線５４２に与える。同様に、セレクタ５３７もこの制御信号によって制御され、命令に対するアクセスのときは信号線５４１を選択し信号線３２を信号線５４１と電気的に接続し、データに対するアクセスのときは信号線５６２を選択し信号線５６２を信号線１０７に接続する。

【００４４】

図６を参照して、トランスレーション回路５４０は、信号線５４１を介してメモリ５２３からデータバス３２の幅にあわせて４バイト単位でフェッチしたＪＡＶＡバイトコードを一時的に蓄えるための入力バッファメモリ５５１と、入力バッファメモリ５５１から与えられるＪＡＶＡバイトコード１バイトを変換して８バイトのネイティブ命令一つを出力する変換器５５２と、変換器５５２の出力、信号線５４５および５４６からのアドレス信号および制御信号を受け、変換器５５２から出力されたネイティブコードを一時蓄えて信号線５３９に出力するための出力バッファメモリ５５３と、信号線５４４、５４５および５４６に接続され

、セレクタ５３５から信号線５４５を介して入力されたアドレスを管理するためのアドレス管理回路５５４とを含む。

【００４５】

図７を参照して、プロセッサ５１０は、命令を以下のようにしてパイプライン処理する。プロセッサ５１０は、機能的には、メモリ演算部１３０と整数演算部１４０とで行なわれるサブ命令をそれぞれ実行するためのＭＵパイプ１３９およびＩＵパイプ１４９を含む。これらパイプはいずれも、命令フェッチステージ１９１、デコードおよびアドレス計算ステージ１９２、演算およびメモリアクセスステージ１９３およびライトバックステージ１９４からなる。

【００４６】

命令フェッチステージ１９１は、命令をフェッチして命令デコーダ１１０中の命令レジスタ１１３に保持するステージである。デコードおよびアドレス計算ステージ１９２では、この命令がサブ命令デコーダ１１１、１１２でデコードされ、同時にレジスタファイル１２０（入力であることを示すためにレジスタファイル１２０ａとする。）がアクセスされてオペランドおよびＰＣのアドレス計算が行なわれる。演算およびメモリアクセスステージ１９３では、整数演算およびデータメモリアクセス処理が行なわれる。ライトバックステージ１９４では、演算結果およびメモリからフェッチされたデータがレジスタファイル１２０（出力側であることを示すためにレジスタファイル１２０ｂとする。）に再び書込まれる。

【００４７】

図８を参照して、プロセッサ５１０の命令２００は２ウェイのＶＬＩＷ型命令であり、図示されるようなフォーマットを有する。すなわち、命令２００は、各サブ命令の実行順序と長いサブ命令を定義するＦＭフィールド２０１ａ、２０１ｂと、サブ命令を格納するＬコンテナ２０５およびＲコンテナ２０６と、各サブ命令の実行条件を指定する条件（ＣＣ）フィールド２０３および２０４とを含む。

【００４８】

条件フィールド２０３および２０４は、制御レジスタ１７０であるＰＳＷ中の

フラグＦ０およびＦ１（Ｆ０ビット１７０ｃおよびＦ１ビット１７０ｄ）の値に依存した条件を指定する。たとえば条件フィールド２０３が「０００」のとき、Ｌコンテナ２０５に含まれるサブ命令は無条件に実行される。条件フィールド２０４が「１０１」のとき、Ｒコンテナ２０６に含まれるサブ命令はＦ０＝１かつＦ１＝１のとき実行され、フラグＦ０およびＦ１（Ｆ０ビット１７０ｃおよびＦ１ビット１７０ｄ）がそれ以外の値をとる場合は無効化される。

【００４９】

ＦＭフィールド２０１ａおよび２０１ｂは、Ｌコンテナ２０５とＲコンテナ２０６とに含まれるサブ命令を実行する場合の実行動作を指定する。実行動作としては４つある。１番目は、Ｌコンテナ２０５とＲコンテナ２０６とに含まれるサブ命令を並列に実行する動作である。２番目は、Ｌコンテナ２０５のサブ命令をまず実行して次にＲコンテナ２０６のサブ命令を実行する動作である。３番目は２番目の動作の逆であり、Ｒコンテナ２０６のサブ命令をまず実行して次にＬコンテナ２０５のサブ命令を実行する動作である。４番目はＬコンテナ２０５とＲコンテナ２０６とに分割して保持された１つの長いサブ命令を実行する動作である。すなわち、ＦＭフィールド２０１ａおよびＦＭフィールド２０１ｂの値によって、上述した４つの動作のいずれかが選択される。

【００５０】

図９を参照して、Ｌコンテナ２０５およびＲコンテナ２０６に保持されるサブ命令は以下のようなフォーマットを有する。サブ命令は２８ビット長の短い命令と５４ビット長の長いサブ命令とに分類される。短いサブ命令は、フォーマット２１１～２１７に示される７種類のフォーマットを有する。短いサブ命令フォーマットの概略をいえば、ビット位置０－９で演算の種類が示され、ビット位置１０－２７で最大３つのオペランドが指定される。長いサブ命令はフォーマット２１８に示されるようにビット位置０－９で演算の種類が示され、ビット位置１０－５３で３２ビット長の即値データを含む最大３つのオペランドが指定される。なお、長いサブ命令の３２ビットの即値はＶＬＩＷ命令ビット位置２６－３１、３６－４３、および４６－６３に保持される。

【００５１】

フォーマット２１１は、メモリアクセス演算（ロード／ストア演算）を行なうサブ命令のフォーマットである。フォーマット２１２は汎用レジスタに保持されたオペランド間の演算（ＯＰ演算）を行なうサブ命令のフォーマットである。フォーマット２１３－２１７は分岐演算を行なうサブ命令のフォーマットである。長いサブ命令のフォーマット２１８は、上記した３種類の演算全てに共通で使用される。

【００５２】

サブ命令を図７に示すようにプロセッサ５１０でパイプライン処理する場合には、ＯＰ演算、ロード／ストア演算、分岐演算のサブ命令はそれぞれ図１０に示す４段のパイプライン２２１－２２３によって示されるように４段のパイプラインステージで実行される。これら各ステージは、図６に示される４つのステージに対応する。

【００５３】

ＦＭフィールド２０１ａおよび２０１ｂによりサブ命令の実行順序が指定されたときには、サブ命令はＭＵパイプ１３９およびＩＵパイプ１４９によって図１１に示すようにパイプライン処理される。ここで、ストールステージ２３４－２３６は、ＦＭフィールド２０１ａおよび２０１ｂの値にしたがってサブ命令に順序をつけて実行する場合に、一方のサブ命令の実行を遅延させるために挿入されるパイプラインステージである。

【００５４】

次に、プロセッサ５１０に対して定義されたサブ命令の一覧を示す。この一覧において、各サブ命令のニーモニックを大文字で、その処理内容を各ニーモニックの右側に、それぞれ示す。

【００５５】

ロード／ストア命令

LDB	符号拡張してレジスタに１バイトをロード
LDBU	ゼロ拡張してレジスタに１バイトをロード
LDH	符号拡張してレジスタに半ワードをロード
LDHH	レジスタ上位に半ワードをロード

LDHU　ゼロ拡張してレジスタに半ワードをロード

LDW　レジスタに１ワードをロード

LD2W　２つのレジスタに２ワードをロード

LD4BH　符号拡張して４つの半ワードレジスタに４つのバイトをロード

LD4BHU　ゼロ拡張して４つの半ワードレジスタに４つのバイトをロード

LD2H　２つのレジスタに２つの半ワードをロード

STB　レジスタから１バイトをストア

STH　レジスタから半ワードをストア

STHH　レジスタ上位から半ワードをストア

STW　レジスタから１ワードをストア

ST2W　レジスタから２ワードをストア

ST4HB　４つの半ワードレジスタから４つのバイトをストア

ST2H　２つのレジスタから２半ワードをストア

MODDEC　５ビット即値によってレジスタ値をデクリメント

MODINC　５ビット即値によってレジスタ値をインクリメント

転送命令

MVFSYS　制御レジスタから汎用レジスタに転送

MVTSYS　汎用レジスタから制御レジスタに転送

MVFACC　アキュムレータから１ワードを転送

MVTACC　２つの汎用レジスタから２ワードをアキュムレータに転送

比較命令

CMPcc　比較

cc= EQ(000),NE(001),GT(010),GE(011),LT(100),

LE(101),PS-both positive(110), NG-both negative(111)

CMPUcc　符号なし比較

cc=GT(010),GE(011),LT(100),LE(101)

算術演算命令

ABS	絶対値
ADD	加算
ADDC	キャリー付加算
ADDHppp	半ワード加算
	ppp=LLL(000),LLH(001),LHL(010),LHH(011),
	HLL(100),HLH(101),HHL(110),HHH(111)
ADDS	第３オペランドの符号をレジスタRbに加算
ADDS2H	２つの半ワードに符号を加算
ADD2H	２対の半ワードを加算
AVG	正の無限大方向の丸めによる平均
AVG2H	正の無限大方向の丸めにより２対の半ワードを平均
JOINpp	２つの半ワードをジョイン
	pp=LL(00),LH(01),HL(10),HH(11)
SUB	減算
SUBB	ボロー付の減算
SUBHppp	半ワードの減算
	ppp=LLL(000),LLH(001),LHL(010),LHH(011),
	HLL(100),HLH(101),HHL(110),HHH(111)
SUB2H	２対の半ワードの減算

論理演算命令

AND	論理AND
OR	論理OR
NOT	論理NOT
XOR	排他的論理OR
ANDFG	フラグの論理AND
ORFG	フラグの論理OR
NOTFG	フラグの論理NOT

XORFG　　フラグの排他的論理OR

シフト演算命令

SRA　　右に算術シフト

SRAHp　　右に半ワードを算術シフト p=L(0),H(1)

SRA2H　　２つの半ワードを右に算術シフト

SRC　　レジスタを連接して右にシフト

SRL　　右に論理シフト

SRLHp　　半ワードを右に論理シフト p=L(0),H(1)

SRL2H　　２つの半ワードを右に論理シフト

ROT　　右ローテイト

ROT2H　　２つの半ワードを右ローテイト

ビット演算命令

BCLR　　ビットをクリア

BNOT　　ビットを反転

BSET　　ビットをセット

BTST　　ビットをテスト

分岐命令

BRA　　分岐

BRATZR　　ゼロなら分岐

BRATNZ　　ゼロでなければ分岐

BSR　　サブルーチンに分岐

BSRTZR　　ゼロならサブルーチンに分岐

BSRTNZ　　ゼロでなければサブルーチンに分岐

DBRA　　遅延分岐

DBRAI　　即値付遅延分岐

DBSR　　サブルーチンへの遅延分岐

DBSRI	サブルーチンへの即値付遅延分岐
DJMP	遅延ジャンプ
DJMPI	即値付遅延分岐
DJSR	サブルーチンへの遅延ジャンプ
DJSRI	サブルーチンへの即値付遅延ジャンプ
JMP	ジャンプ
JMPTZR	ゼロならジャンプ
JMPTNZ	ゼロでなければジャンプ
JSR	サブルーチンに分岐
JSRTZR	ゼロならサブルーチンにジャンプ
JSRTNZ	ゼロでなければサブルーチンにジャンプ
NOP	ノーオペレーション

OS関連命令

TRAP	トラップ
REIT	例外、割込およびトラップからの復帰

DSP算術演算命令

MUL	乗算
MULX	拡張精度の乗算
MULXS	拡張精度での乗算および左へのシフト
MULX2H	拡張精度で２対の半ワードを乗算
MULHXpp	拡張精度で２つの半ワードを乗算 pp=LL(00),LH(01),HL(10),HH(11)
MUL2H	２対の半ワードを乗算
MACd	乗算および加算（d=0,1）
MACSd	乗算、左１ビットシフト、加算（d=0,1）
MSUBd	乗算および減算（d=0,1）
MSUBSd	乗算、左１ビットシフト、減算（d=0,1）

SAT	サチュレート
SATHH	ワードオペランドを上位半ワードにサチュレート
SAHL	ワードオペランドを下位半ワードにサチュレート
SATZ	正数にサチュレート
SATZ2H	２つの半ワードを正数にサチュレート
SAT2H	２つの半ワードオペランドをサチュレート

繰返し命令

REPEAT	１ブロックの命令を繰返す
REPEATI	１ブロックの命令を即値付繰返す

デバッガサポート命令

DBT	デバッグトラップ
RTD	デバッグ割込およびトラップからの復帰

図１２に、図１に示すプロセッサ５１０がアドレスバス３１、データバス３２および制御バス３３を介して外部の各メモリ５２０～５２３をアクセスする場合のアドレスマップを示す。図１２に示されるように、メモリ５２０～５２２はそれぞれ、ＲＡＭ用アドレス領域５７５(H’ 2220 0000-H’ 223F FFF8)、データメモリ用アドレス領域５７１(H’ 1000 0000-H’ 17FF FFF8)、ネイティブ命令メモリ用アドレス領域５７２(H’ 1800 0000-H’ 1FFF FFF8)にマップされる。メモリ５２３は、命令コードの変換を行なわずにリードライトする場合にはアドレス領域５７４(H’ 2100 0000-H’ 211F FFF8)にマップされ、命令コードの変換を伴ってリードする場合はアドレス領域５７３ (H’ 2000 0000-H’ 20FF FFF8)にマップされる。

【００５６】

プロセッサ５１０から見ると、メモリ５２３のアドレスは図１２に示すアドレス領域５７３およびアドレス領域５７４となる。アドレス領域５７４でメモリ５２３をアクセスする場合、２ＭＢの空間の全バイト位置に有効なメモリがある。

一方、アドレス領域５７３でメモリ５２３をアクセスする場合、１６ＭＢの空間の整地された各８バイトの上位１バイトには有効なメモリがあるが、下位７バイトにはメモリは存在しない。

【００５７】

なお、図５を参照して述べたことから明らかなように、外部のメモリは４バイトに整地されたアドレスによりアクセスされる。そのため外部のデータバス３２は４バイト幅（３２ビット幅）となる。したがって、アドレス領域５７２などをアクセスして命令メモリ５２２から一つのネイティブ命令（８バイトのＶＬＩＷ命令）を読む場合、そのネイティブ命令のアクセスは２回に分けて行なわれる。具体的には、プロセッサがアドレスバス３１に２回連続してアドレスを出力することにより、データバス３２から４バイトのデータが２回連続して受取られる。バス幅変換回路５６０は、４バイト幅の信号線５４１から受ける４バイトのデータを８バイトのデータに変換して８バイト幅の信号線５６３に転送するためのものである。すなわち、バス幅変換回路５６０は、１回目の４バイトデータを信号線５４１から受取り一旦保持し、２回目の４バイトデータを信号線５４１から受取ると、１回目の４バイトデータと合わせた８バイトデータをネイティブ命令として出力する。

【００５８】

バス幅変換回路５６１の機能も同様である。すなわち、アドレス領域５７１などをアクセスして８バイトのデータを読出または書込みする場合、４バイト幅の信号線５６２から２回の連続したメモリアクセスにより受ける２つの４バイトデータを１つの８バイトデータにしてデータバス１０７に出力し、また、データバス１０７から受ける１つの８ビットデータを２つの４ビットデータに分けて２回の連続したメモリアクセスにより信号線５６２、セレクタ５３７を介してデータバス３２に出力する。

【００５９】

なお、もしデータバス３２のバス幅が６４ビットであれば、バス幅変換回路５６０、５６１は不要である。この場合、信号線５４１および５６３は直接に接続される。同様に信号線５６２およびデータバス１０７は直接に接続される。

【００６０】

上に構造を述べてきたプロセッサ５１０は以下のように動作する。最初に命令またはデータのアクセスがアドレス空間５７１～５７２、５７４～５７５に対して行なわれるＨＷトランスレーションなしのメモリアクセス時について説明する。この場合、図１を参照して、制御バス３３上の信号により、アクセス対象となるメモリのチップセレクト信号をアサートする。図５に示すセレクタ５３５および５３８で信号線５４２上の信号を選択してアドレスバス３１にアドレスの下位２４ビットを出力する。このとき、アクセスが命令リードか、データリードか、またはデータライトかに依存して、セレクタ５３６～５３８が制御され命令バス１０５、データバス１０７およびデータバス３２の接続関係とデータの流れが制御される。その結果、トランスレーション回路５４０の出力ではなく、セレクタ５３７の出力がセレクタ５３６によって選択され命令として信号線１０５を介してプロセッサコア１００に与えられる。

【００６１】

命令アクセスがアドレス空間５７３に行なわれる、ＨＷトランスレーション付きのメモリアクセス時は、制御部５３２が信号線５４６を介してトランスレーション回路５４０、セレクタ５３５および５３６を制御し、メモリ５２３からフェッチしたＪＡＶＡバイトコードをネイティブ命令に変換して命令バス１０５上に出力する。このとき、命令アドレスバス１０４からセレクタ５３１を介して信号線５４２に、さらにアドレス信号シフト回路５３４に入力されたアドレス値は、アドレス信号シフト回路５３４によって３ビット右にシフトされる。すなわちアドレス値は１／８の値となって信号線５４５上に出力され、トランスレーション回路５４０に入力される。

【００６２】

図６を参照して、トランスレーション回路５４０のアドレス管理回路５５４は、信号線５４５を介して入力された、アドレス領域５７３のアドレス値をアドレス領域５７４にマップしなおして信号線５４４上に出力する。この信号が図５に示されるセレクタ５３８を介してアドレスバス３１上に出力される。その結果メモリ５２３がアクセスされ、ＪＡＶＡバイトコードがフェッチされる。

【００６３】

このときのトランスレーション回路５４０の動作の詳細を図６を参照して説明する。アドレス管理回路５５４は、信号線５４５から入力されたアドレス値を、直前に入力されたアドレス値と比較し、対応するＪＡＶＡバイトコードが入力バッファメモリ５５１内に存在しているか判定する。もし対応するＪＡＶＡバイトコードが存在しているときにはそのＪＡＶＡバイトコードを入力バッファメモリ５５１から変換器５５２に出力させる。

【００６４】

対応するＪＡＶＡバイトコードが入力バッファメモリ５５１内に存在していない場合、アドレス管理回路５５４は、信号線５４５を介して入力された、アドレス領域５７３の下位２４ビットのアドレス値をアドレス領域５７４の下位２４ビットのアドレス値としてメモリ５２３をアクセスするための制御信号を制御信号線５４６上に出力する。この際、ＪＡＶＡバイトコードを整地された４バイト単位にフェッチするため、アドレス管理回路５５４から信号線５４４上に出力される２４ビットのアドレス値の下位２ビットは、入力されたアドレス値にかかわらず常に０である。このアドレス値は図５に示すようにセレクタ５３８によって選択され、アドレスバス３１上に出力される。これにより、ＪＡＶＡバイトコードは４バイト単位でメモリ５２３からデータバス３２を通して入力バッファメモリ５５１にフェッチされる。

【００６５】

図１３を参照して、命令コード変換部３７０が変換するＪＡＶＡ命令とＶＬＩＷ命令との関係は以下のとおりである。たとえば１バイトのＪＡＶＡ命令４０１は８バイトのＶＬＩＷ命令４１１に変換される。ＪＡＶＡ命令４０１は１バイトなので、このＪＡＶＡ命令４０１はＪＡＶＡ命令用メモリ５２３を１回アクセスするだけで読出すことができる。変換後のＶＬＩＷ命令４１１は出力バッファ５５３を経由して信号線５３９上に出力される。今度はセレクタ５３６は、アドレスがアドレス領域５７３の内部であるので、信号線５３９上の命令を選択してプロセッサコア１００に与える。

【００６６】

２バイトのＪＡＶＡ命令４０２は２つの８バイトのＶＬＩＷ命令４１２ａ、４１２ｂに変換される。５バイトのＪＡＶＡ命令４０４は３つの８バイトのＶＬＩＷ命令４１４ａ、４１４ｂおよび４１４ｃに変換される。ＪＡＶＡ命令４０３、ＪＡＶＡ命令４０６～４０８などについても同様で、ＪＡＶＡ命令４０３はＶＬＩＷ命令４１３に、ＪＡＶＡ命令４０６はＶＬＩＷ命令４１６に、ＪＡＶＡ命令４０７はＶＬＩＷ命令４１７に、ＪＡＶＡ命令４０８は３つの８バイトのＶＬＩＷ命令４１８ａ～４１８ｃに、それぞれ変換される。

【００６７】

図１４に、コア１００がメモリ領域５７３のアドレスで命令アクセス要求をしてからメモリ５２３がアクセスされてＪＡＶＡバイトコードがフェッチされ、ネイティブ命令に変換されてコア１００へ転送される間のデータの流れを示す。コア１００がメモリ領域５７３のアクセスを行なうときは、これは必ずネイティブ命令のフェッチである。したがってコア１００は常に８バイトに整地されたアドレスをバスインターフェース部５０３に出力する。

【００６８】

メモリ領域５７３のアドレスはバスインターフェース部５０３でメモリ領域５７４のアドレスとして出力されメモリ５２３に対するアクセスが行なわれる。メモリ領域５７３は仮想的なＪＡＶＡバイトコード領域である。そのため、実際のメモリ５２３をアクセスしてフェッチされたＪＡＶＡバイトコード（非ネイティブ命令）はバスインターフェース部５０３でネイティブ命令に変換されてバス１０５からコア１００へ転送される。つまり、コア１００から見るとメモリ領域５７３の１６ＭＢのネイティブ命令が１／８に圧縮され、メモリ領域５７４の２ＭＢのＪＡＶＡバイトコードとしてメモリ５２３に保存されている。そしてバスインターフェース部５０３がメモリ５２３から４バイト単位でフェッチしたＪＡＶＡバイトコードを一旦入力バッファ５５１に蓄え、さらに入力バッファ５５１から変換器５５２に読出してネイティブ命令への変換を行なって出力バッファメモリ５５３を介してコア１００に出力することで命令コードを伸長する機能を実現する。

【００６９】

なお、アドレス領域５７４をアクセスするときは、プロセッサ５１０はＪＡＶＡ命令メモリ５２３に記憶されたＪＡＶＡバイトコードをデータとしてアクセスすることができる。したがってフェッチすべきＪＡＶＡバイトコードを指定するアドレスはデータアドレスバス１０６を介してアドレスバス３１に転送され、ＪＡＶＡバイトコードはデータバス３２を介してメモリ５２３からデータバス１０７へ転送される。プロセッサ５１０は、「ソフトウェアインタープリタ」と呼ばれるネイティブ命令で記述されたソフトウェアを用いて、データとしてフェッチしたＪＡＶＡバイトコードを実行することができ、または「ソフトウェアトランスレータ」と呼ばれるネイティブ命令で記述されたソフトウェアを用いてデータとしてフェッチしたＪＡＶＡバイトコードをネイティブ命令に変換し、その後実行することができる。したがって本実施の形態によるプロセッサ５０１は、ハードウェアトランスレータ、ソフトウェアトランスレータ、およびソフトウェアインタープリタのいずれの方法によっても、ＪＡＶＡバイトコードのような非ネイティブ命令で記述されたプログラムを実行できるようサポートされている。

【００７０】

なおプロセッサがどのような基準にしたがってハードウェアトランスレータ、ソフトウェアトランスレータ、およびソフトウェアインタープリタのいずれかを選択してＪＡＶＡバイトコードを実行するかについては、たとえば２０００年１２月４日出願の、本願出願人と同一出願人による特願２０００－３６８７２９号において検討されている。

【００７１】

図１５～図２０は、変換器３７２でＪＡＶＡ命令をＶＬＩＷ命令に変換したときの具体的な例を示す。なお、ＪＡＶＡ命令についてはTim Lindholm and Frank Yellin, "The Java Virtual Machine Specification Second Edition," Sun Microsystems, Inc., 1999に詳しく記載されている。

【００７２】

図１５を参照して、１バイトのＪＡＶＡ命令「iadd」はサブ命令「LDW R61, @(R63+, R0)」とサブ命令「ADD R62, R62, R61」とを順に実行する１つのＶＬＩＷ命令に変換される。「iadd」はスタックトップから１番目のデータと２番目の

データとである２つの３２ビット整数を加算してスタックに書き戻すＪＡＶＡ命令である。プロセッサ５１０ではスタックトップのデータはレジスタＲ６２に配置され、レジスタＲ６３がスタックトップの次のデータのアドレスを示す。したがって、プロセッサ５１０では、ＪＡＶＡ命令「iadd」のオペレーションを、スタックトップから２番目の３２ビットデータをレジスタＲ６１にロードしてレジスタＲ６２を４インクリメントするオペレーションを行なうサブ命令「LDW R61, @(R63+, R0)」と、レジスタＲ６２とレジスタＲ６１との２つの３２ビット整数を加算してその結果をレジスタＲ６２に書込むオペレーションを行なうサブ命令「ADD R62, R62, R61」とでエミュレートすることができる。このとき、ＰＣ値については、１つのＶＬＩＷ命令を実行したことによりプロセッサ５１０のＰＣ値を８番地進めることで、ＪＡＶＡ命令「iadd」の実行に対応してＰＣ値を１進めることをエミュレートする。

【００７３】

図１６を参照して、２バイトのＪＡＶＡ命令「iload」は、１つのサブ命令「ADD/CN R50, #(0||vindex)」を持つＶＬＩＷ命令と、サブ命令「STW R62, @(R63-, R4)」とサブ命令「LDW R62, @(R10, R50)」とを順次に実行するＶＬＩＷ命令に変換される。「iload」はローカル変数領域から３２ビット整数をフェッチしてスタックトップに保存するＪＡＶＡ命令である。プロセッサ５１０ではこれを、レジスタＲ５０にローカル変数のインデックス値をロードするサブ命令「ADD/CN R50, #(0||vindex)」と、レジスタＲ６２をスタックトップから２番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、スタックトップであるレジスタＲ６２にローカル変数領域からデータをロードするサブ命令「LDW R62, @(R10, R50)」とに分解することによりシミュレートする。

【００７４】

なお、ここでレジスタＲ４には、値「－４」を保持し、レジスタＲ１０にはローカル変数領域のベースアドレスを保持する。ＰＣ値については２つのＶＬＩＷ命令を実行したことによりプロセッサ５１０のＰＣ値を１６番地進めることで、ＪＡＶＡ命令「iload」の実行に対応してＰＣ値を２番地進めることをシミュレートする。

【００７５】

図１７を参照して、３バイトのＪＡＶＡ命令「ifeq」は、サブ命令「ADD R62, R61, R0」とサブ命令「NOP」とを並列に実行するＶＬＩＷ命令と、サブ命令「LDW R62, @(R63+, R0)」とサブ命令「NOP」とを並列に実行するＶＬＩＷ命令と、１つのサブ命令「BRATZR/CN R62, #(s||branchbyte1||branchbyte2)」を実行するＶＬＩＷ命令とに分解される。命令「ifeq」はスタックトップのデータが「０」なら分岐するＪＡＶＡ命令である。プロセッサ５１０ではこれを、レジスタＲ６２をレジスタＲ６１にコピーするサブ命令「ADD R62, R61, R0」と、スタックトップから２番目のデータをレジスタＲ６２にポップするサブ命令「LDW R62, @(R63+, R0)」と、レジスタＲ６１がゼロなら分岐するサブ命令「BRATZR/CN R62, #(s||branchbyte1||branchbyte2)」とに分解し、これら３つのサブ命令と２つのNOP命令とを組合せて、計３つのＶＬＩＷ命令でシミュレートする。ＰＣ値については、３つのＶＬＩＷ命令を実行したことによりプロセッサ５１０のＰＣ値を２４番地進めることで、ＪＡＶＡ命令「ifeq」の実行に対応してＰＣ値を３番地進めることをシミュレートする。

【００７６】

図１８を参照して、５バイトのＪＡＶＡ命令「jsr_w」は、１つのサブ命令「OR R10, #(branchbyte1||branchbyte2||branchbyte3||branchbyte4)」からなるＶＬＩＷ命令と、サブ命令「STW R62, @(R63-, R4)」とサブ命令「BSR R10」とを順次に実行するＶＬＩＷ命令と、サブ命令「BRA #3」とサブ命令「NOP」とを並列に実行するＶＬＩＷ命令とに変換される。「jsr_w」は戻り先アドレスをスタックにプッシュして４バイトで指定されたアドレスのサブルーチンへジャンプするＪＡＶＡ命令である。プロセッサ５１０ではこの命令を、ジャンプ先アドレスをレジスタＲ１０にロードするサブ命令「OR R10, #(branchbyte1||branchbyte2||branchbyte3||branchbyte4)」と、スタックトップのレジスタＲ６２の値をスタックトップから２番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタＲ６２に保存してレジスタＲ１０で指定されたアドレスのサブルーチンへジャンプするサブ命令「JSR R10」と、サブルーチンから復帰した後に２つのＶＬＩＷ命令をスキップするための分

岐を行なうサブ命令「BRA #3」と、サブ命令「NOP」とに分解してシミュレートする。

【００７７】

ＰＣ値については、３つのＶＬＩＷ命令を実行し、２つのＶＬＩＷ命令をスキップする分岐を行なうことによりプロセッサ５１０のＰＣ値を４０番地進めることで、ＪＡＶＡ命令「jsr_w」の実行に対応してＰＣ値を５番地進めることをシミュレートする。

【００７８】

図１９と図２０とは、複雑なＪＡＶＡ命令を、ＶＬＩＷ命令からなりかつＪＡＶＡ命令の機能を実行するサブルーチンをコールするＶＬＩＷ命令に変換する例を示す。図１９を参照して、この例では、浮動小数点数の加算を行なうＪＡＶＡ命令「fadd」を、スタックトップのレジスタＲ６２の値をスタックトップから２番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタＲ６２に保存して#faddで指定されたアドレスのサブルーチンにジャンプするサブ命令「JSR #fadd」とを順次に実行するＶＬＩＷ命令に変換する。プロセッサ５１０では、「fadd」のオペレーションをサブルーチン中で行ない、ＰＣ値の更新に関しては、一つのＶＬＩＷ命令を実行することでプロセッサ５１０のＰＣ値を８番地進めて「fadd」に対応するＰＣ値を１番地進めることをシミュレートする。

【００７９】

図２０に示す例では、テーブルジャンプを行なうＪＡＶＡ命令「tableswitch」を、スタックトップのレジスタＲ６２の値をスタックトップから２番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタＲ６２に保存して#tableswitchで指定されたアドレスのサブルーチンへジャンプするサブ命令「JSR #tableswitch」とを順次に実行するＶＬＩＷ命令に変換する。プロセッサ５１０では、「tableswitch」のオペレーションとＰＣ値の更新との両方をサブルーチン中でＪＡＶＡ命令「tableswitch」で指定された各種パラメータをアクセスしてシミュレートする。このとき、プロセッサ５１０は、ＪＡＶＡ命令用（直結）アドレス領域１２５ａでＪＡＶＡ命令用の

トランスレータ付メモリ２５をアクセスしてＪＡＶＡ命令「tableswitch」で指定された各種パラメータをデータとして読出す。

【００８０】

今回開示された実施の形態はすべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【００８１】

【発明の効果】

請求項１に記載の発明によれば、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを高速に実行することができる。

【００８２】

請求項２に記載の発明によれば、請求項１に記載の発明の効果に加えて、非ネイティブ命令の命令長とネイティブ命令の命令長とが異なる場合でも、プロセッサコアは命令長の違いを意識する必要がなく、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを実行することが可能となる。

【００８３】

請求項３に記載の発明によれば、請求項２に記載の発明の効果に加えて、非ネイティブ命令の命令長がネイティブ命令の命令長を２のｎ乗で除した大きさのとき、単純な回路だけで非ネイティブ命令をフェッチするためのアドレス変換を行なうことができる。

【００８４】

請求項４に記載の発明によれば、請求項３に記載の発明の効果に加えて、非常に単純な回路で非ネイティブ命令をフェッチするためのアドレス変換が行なえる。

【００８５】

請求項５に記載の発明によれば、このメモリインタフェースを有するプロセッサコアは実行中の命令がネイティブ命令か、非ネイティブ命令かを判断すること

なく命令を単純に実行することにより、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを実行することができる。プロセッサコア内部ではネイティブ命令のみなので、プロセッサコアにおける実行を高速にすることができる。

【００８６】

請求項６に記載の発明によれば、請求項５に記載の発明の効果に加えて、非ネイティブ命令の命令長とネイティブ命令の命令長とが異なる場合でも、プロセッサコアは命令長の違いを意識する必要がなく、プロセッサの構成に変更を加えずに非ネイティブ命令の処理ルーチンとネイティブ命令の処理ルーチンとが混在したプログラムを実行することができる。

【００８７】

請求項７に記載の発明によれば、請求項６に記載の発明の効果に加えて、非ネイティブ命令の命令長がネイティブ命令の命令長を２のｎ乗で除した大きさのとき、単純な回路で非ネイティブ命令をフェッチするためのアドレス変換を行なうことができる。

【００８８】

請求項８に記載の発明によれば、請求項７に記載の発明の効果に加えて、非常に単純な回路で非ネイティブ命令をフェッチするためのアドレス変換が行なえる。

【図面の簡単な説明】

【図１】　本発明の１実施の形態にかかる命令トランスレータ機能付メモリを備えたデータ処理装置のブロック図である。

【図２】　図１に示すプロセッサ５１０のブロック図である。

【図３】　プロセッサ５１０が有するレジスタの一覧を表形式で示す図である。

【図４】　プロセッサ５１０の制御レジスタ１７０の詳細を示す図である。

【図５】　プロセッサ５１０のバスインターフェース部５０３のブロック図である。

【図６】　バスインターフェース部５０３のトランスレーション回路５４０

のブロック図である。

【図７】　プロセッサ５１０のパイプライン処理機構を説明するための図である。

【図８】　プロセッサ５１０で実行可能なＶＬＩＷ命令のフォーマットを示す図である。

【図９】　プロセッサ５１０で実行可能なＶＬＩＷ命令のサブ命令のフォーマットを示す図である。

【図１０】　プロセッサ５１０のサブ命令のパイプライン処理方法を示すための図である

【図１１】　プロセッサ５１０で２つのサブ命令をパイプライン処理する方法を説明するための図である。

【図１２】　図１に示すデータ処理装置のメモリマップを示す図である。

【図１３】　ＪＡＶＡ命令を１または複数のＶＬＩＷ命令に変換するときの命令間の対応を示す図である。

【図１４】　メモリ５２３からＪＡＶＡバイトコードがフェッチされ、ネイティブ命令に変換されてコア１００へ転送される間のデータの流れを示す図である。

【図１５】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図１６】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図１７】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図１８】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図１９】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

【図２０】　ＪＡＶＡ命令をＶＬＩＷ命令に変換する具体例を示す図である。

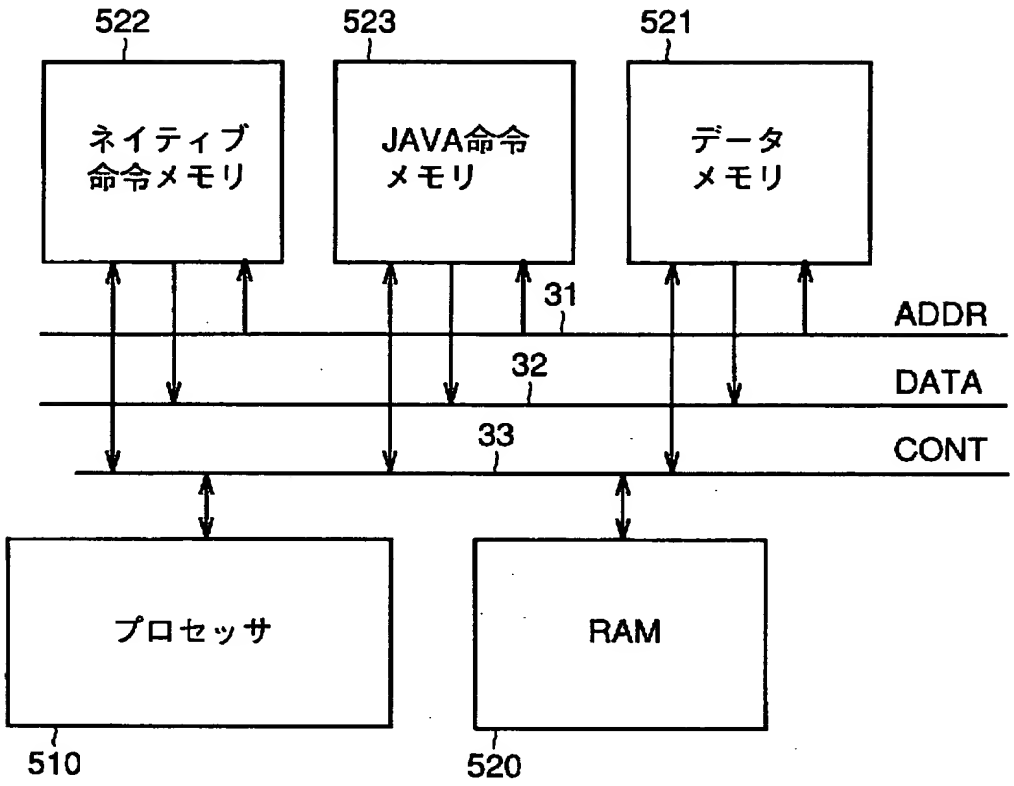
【図２１】　従来の旧命令エミュレーション機能付データ処理装置の概略ブロック図である。

【符号の説明】

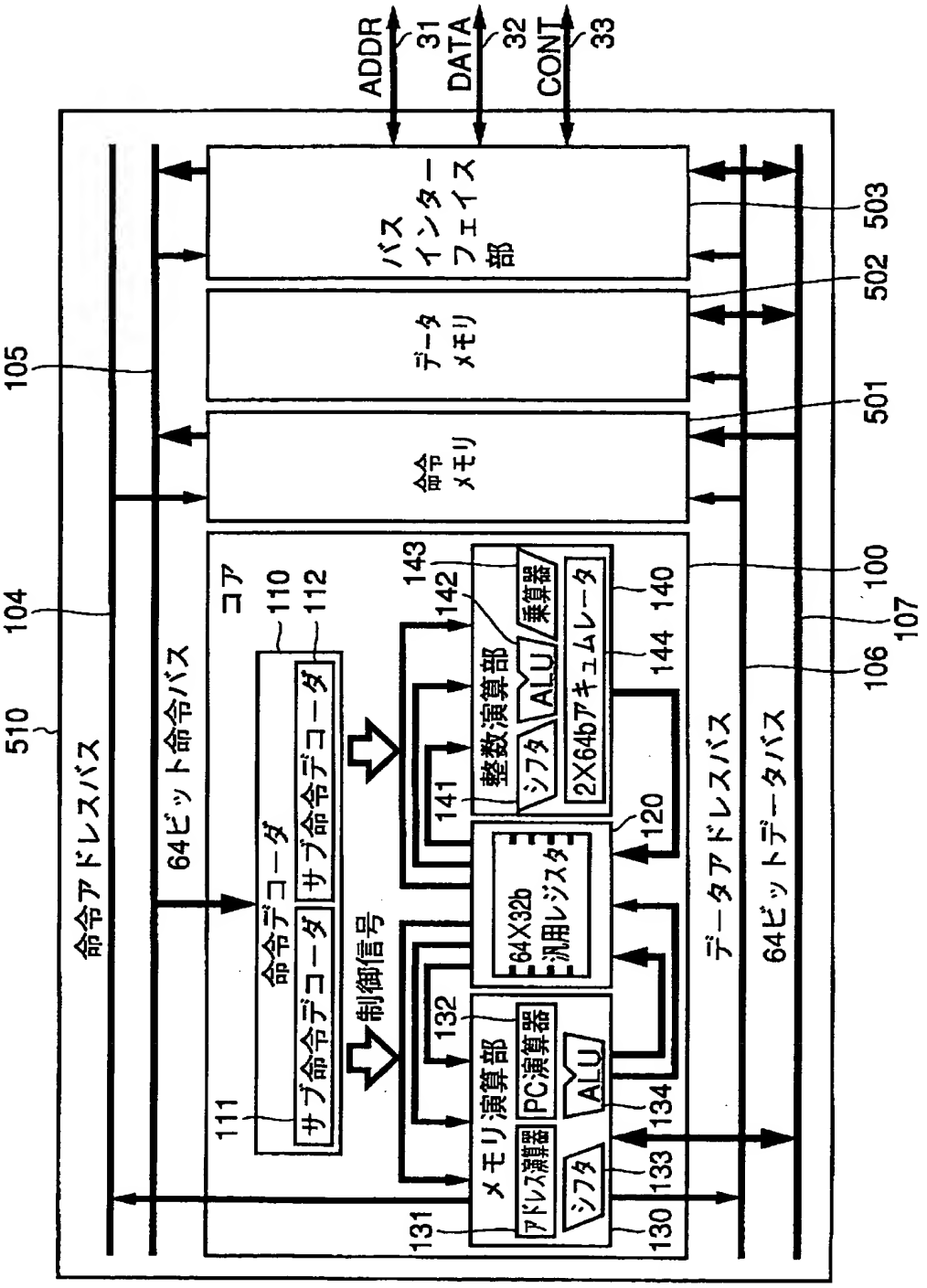
５１０　プロセッサ、５２０　ＲＡＭ、５２１　データメモリ、５２２　ネイティブ命令メモリ、５２３　ＪＡＶＡ命令メモリ、５０３　バスインターフェース部、５４０　トランスレーション回路、５５４　アドレス管理回路、５５２　変換器。

【書類名】　　図面

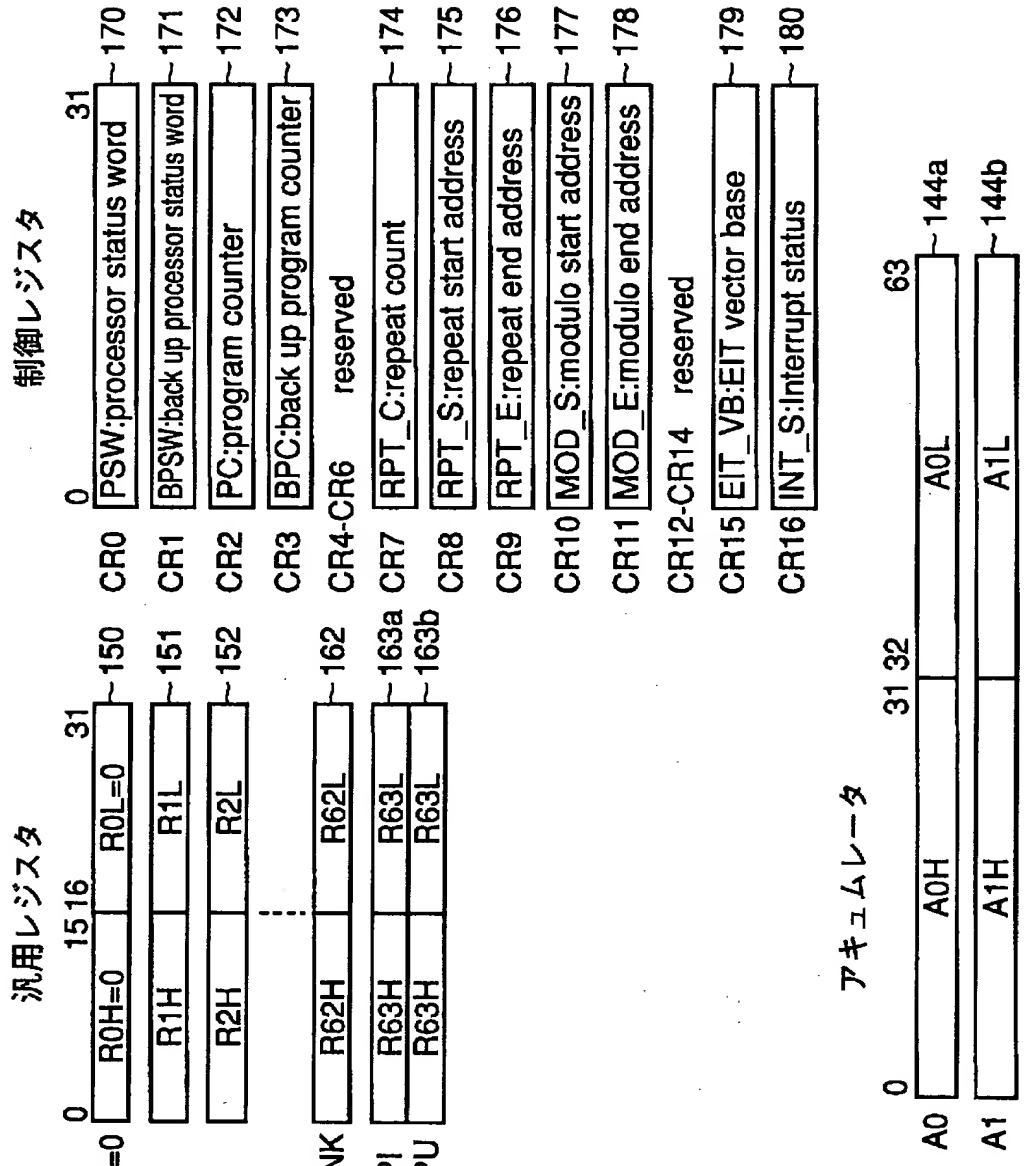
【図１】


522
523
521
ネイティブ
命令メモリ
JAVA命令
メモリ
データ
メモリ
31
ADDR
32
DATA
33
CONT
プロセッサ
RAM
510
520

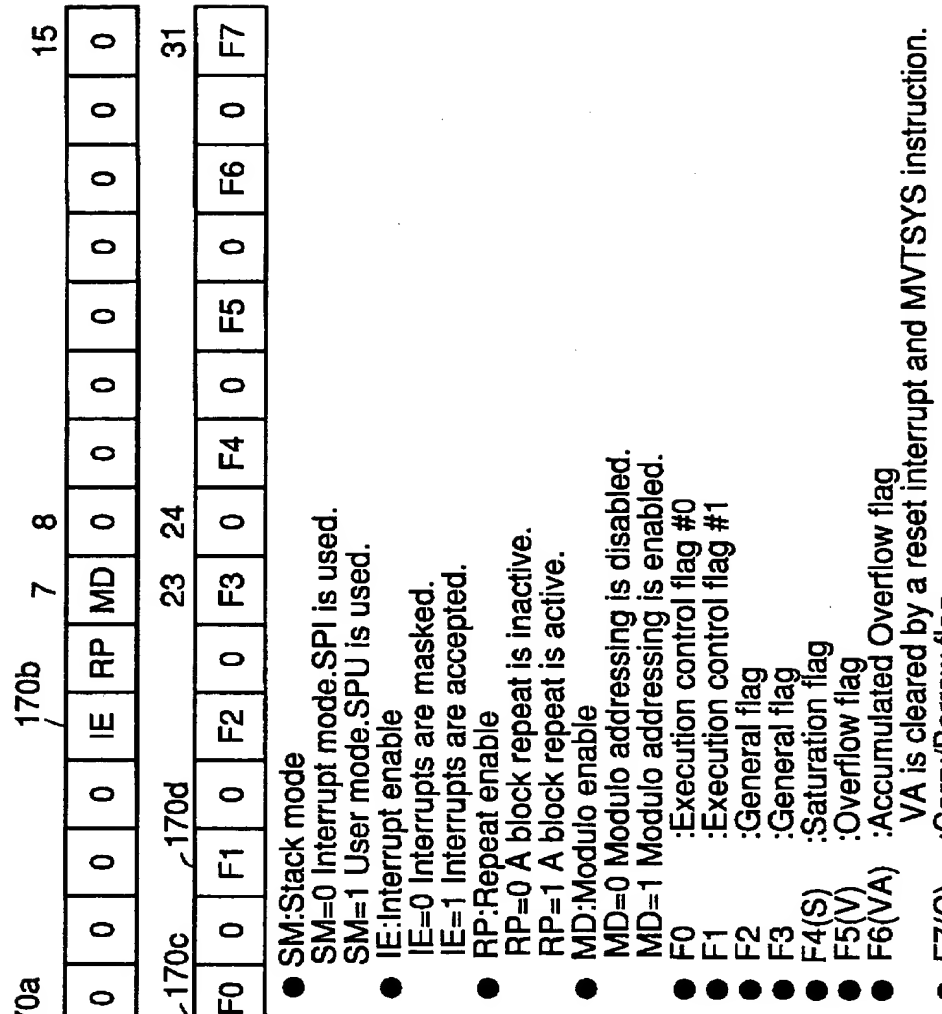
【図２】


ADDR
31
DATA
32
CONT
33
バス インターフェイス 部
503
データ メモリ
502
命令 メモリ
501
105
104
510
命令アドレスバス
64ビット命令バス
コア
110
112
命令デコーダ
サブ命令デコーダ
サブ命令デコーダ
111
制御信号
143
142
整数演算部
シフタ
ALU
乗算器
2X64bアキュムレータ
140
144
100
107
106
データアドレスバス
64ビットデータバス
141
120
64X32b 汎用レジスタ
132
メモリ演算部
アドレス演算器
PC演算器
ALU
シフタ
134
133
131
130

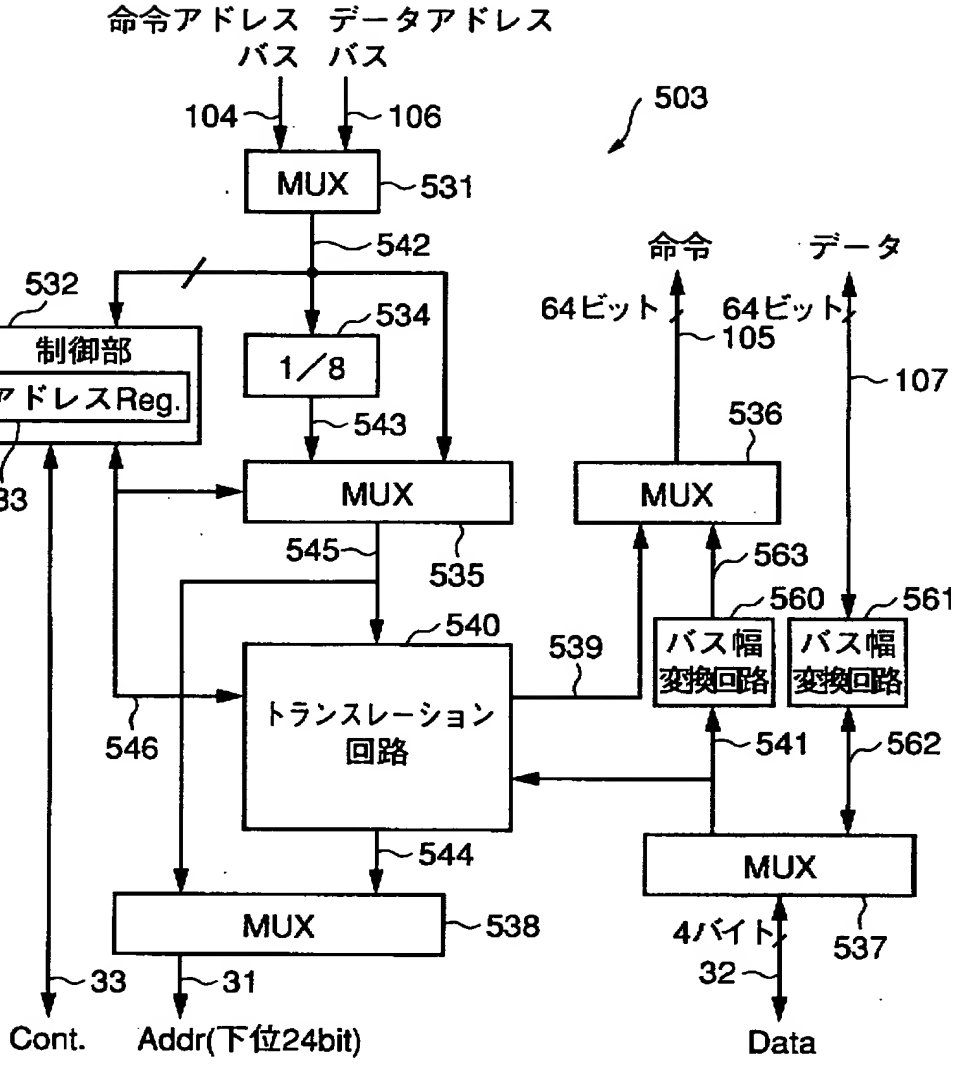
【図３】


制御レジスタ
汎用レジスタ
アキュムレータ
PSW:processor status word
BPSW:back up processor status word
PC:program counter
BPC:back up program counter
reserved
RPT_C:repeat count
RPT_S:repeat start address
RPT_E:repeat end address
MOD_S:modulo start address
MOD_E:modulo end address
reserved
EIT_VB:EIT vector base
INT_S:Interrupt status
R0H=0
R0L=0
R1H
R1L
R2H
R2L
R62H
R62L
R63H
R63L
A0H
A0L
A1H
A1L
R0=0
R1
R2
R62=LINK
R63=SPI
R63=SPU
A0
A1
CR0
CR1
CR2
CR3
CR4-CR6
CR7
CR8
CR9
CR10
CR11
CR12-CR14
CR15
CR16
150
151
152
162
163a
163b
170
171
172
173
174
175
176
177
178
179
180
144a
144b

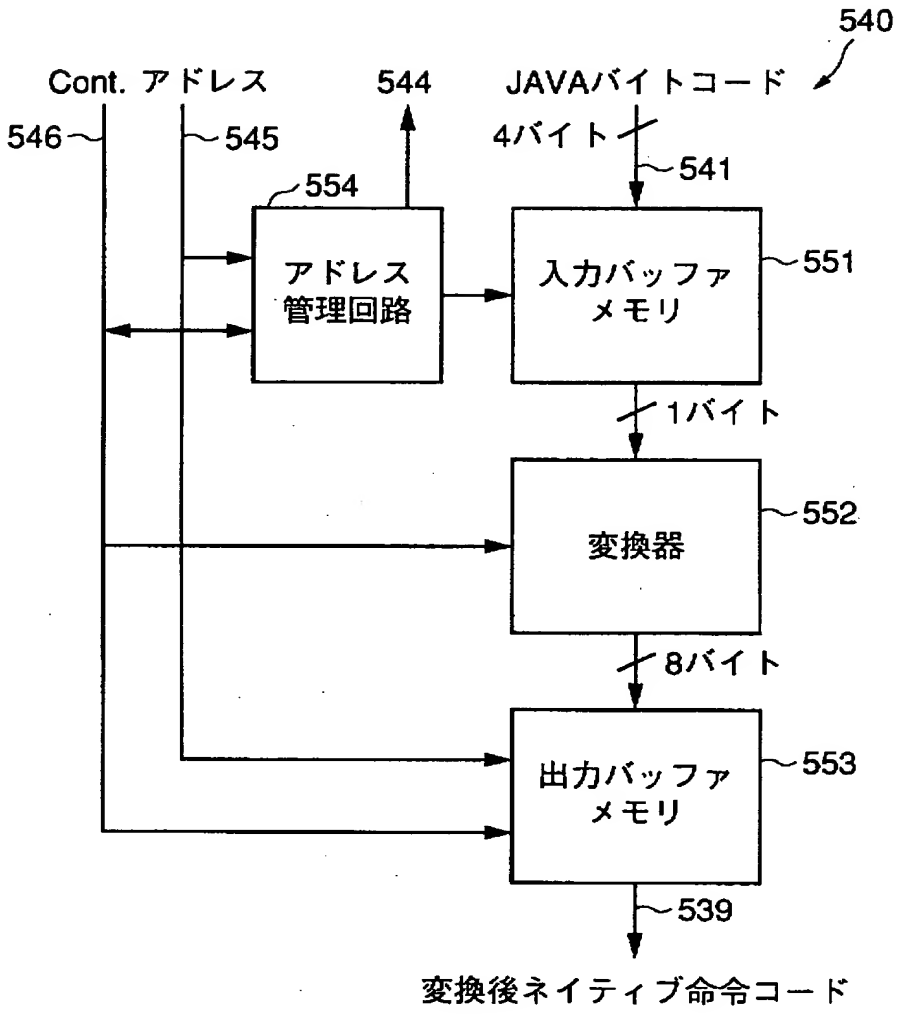
【図４】


0
170a
170b
7
8
15
SM
0
0
0
0
IE
RP
MD
0
0
0
0
0
0
0
16
170c
170d
23
24
31
0
F0
0
F1
0
F2
0
F3
0
F4
0
F5
0
F6
0
F7
● SM:Stack mode
SM=0 Interrupt mode.SPI is used.
SM=1 User mode.SPU is used.
● IE:Interrupt enable
IE=0 Interrupts are masked.
IE=1 Interrupts are accepted.
● RP:Repeat enable
RP=0 A block repeat is inactive.
RP=1 A block repeat is active.
● MD:Modulo enable
MD=0 Modulo addressing is disabled.
MD=1 Modulo addressing is enabled.
● F0 :Execution control flag #0
● F1 :Execution control flag #1
● F2 :General flag
● F3 :General flag
● F4(S) :Saturation flag
● F5(V) :Overflow flag
● F6(VA) :Accumulated Overflow flag
VA is cleared by a reset interrupt and MVTSYS instruction.
● F7(C) :Carry/Borrow flag

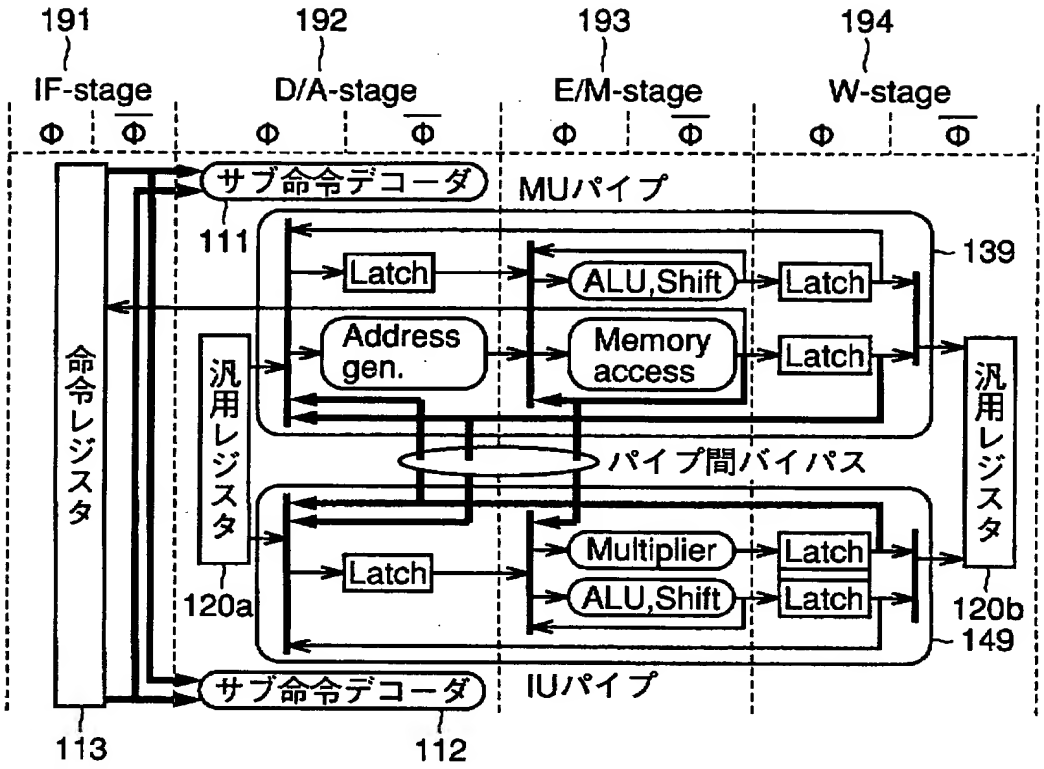
【図５】


命令アドレス バス
データアドレス バス
104
106
503
MUX
531
542
命令
データ
532
534
64ビット
64ビット
105
107
制御部
アドレスReg.
1／8
536
543
533
MUX
MUX
545
535
563
560
561
540
539
バス幅 変換回路
バス幅 変換回路
546
トランスレーション 回路
541
562
544
MUX
MUX
538
4バイト
537
32
33
31
Cont.
Addr(下位24bit)
Data

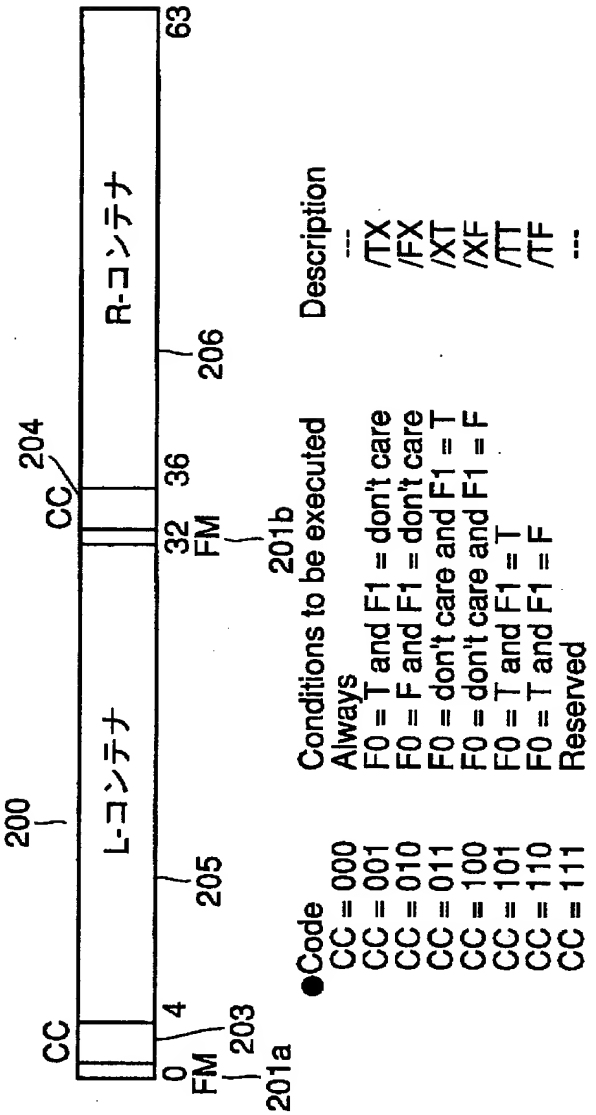
【図６】


540
Cont. アドレス
544
JAVAバイトコード
546
545
4バイト
541
554
アドレス
管理回路
入力バッファ
メモリ
551
1バイト
変換器
552
8バイト
出力バッファ
メモリ
553
539
変換後ネイティブ命令コード

【図７】

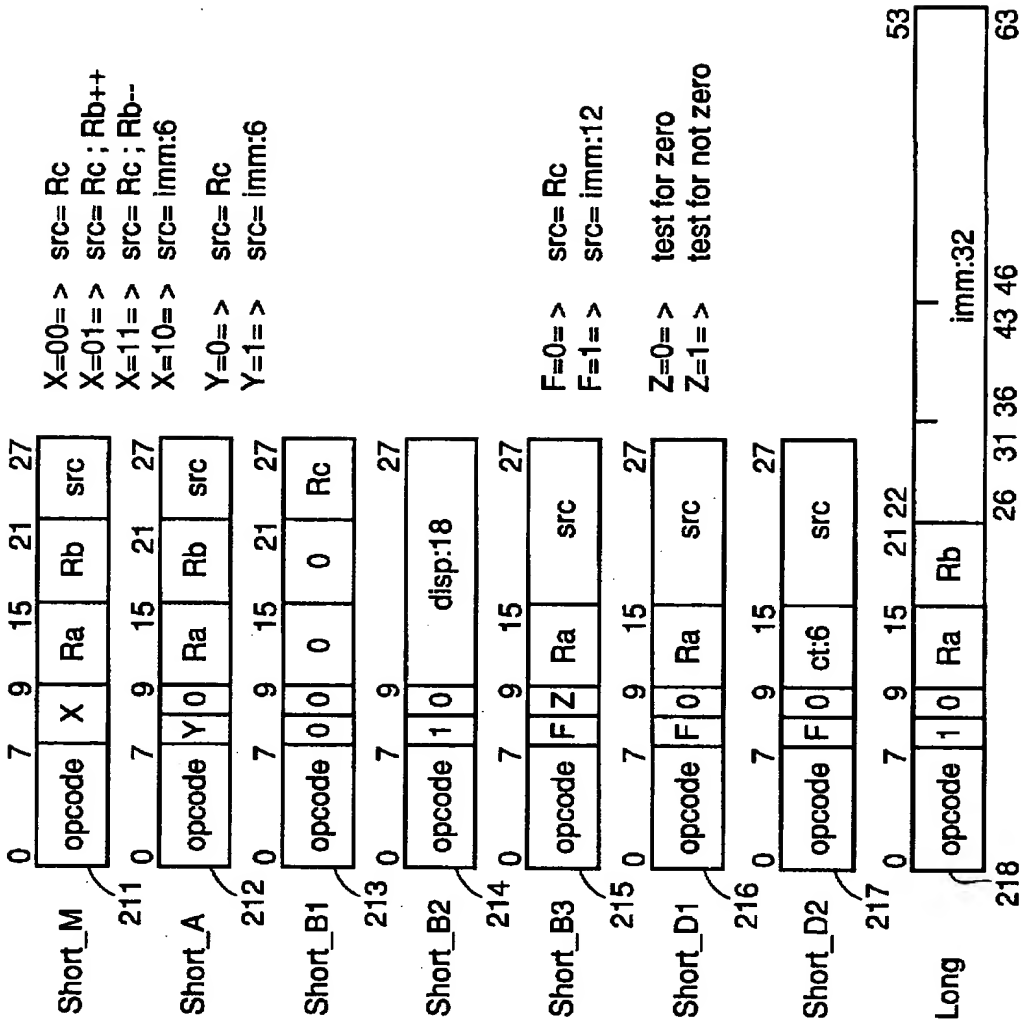

191
192
193
194
IF-stage
D/A-stage
E/M-stage
W-stage
Φ
Φ
Φ
Φ
Φ
Φ
Φ
Φ
サブ命令デコーダ
MUパイプ
111
139
Latch
ALU,Shift
Latch
Address gen.
Memory access
Latch
命令レジスタ
汎用レジスタ
汎用レジスタ
パイプ間バイパス
Multiplier
Latch
Latch
ALU,Shift
Latch
120a
120b
149
サブ命令デコーダ
IUパイプ
113
112

【図８】

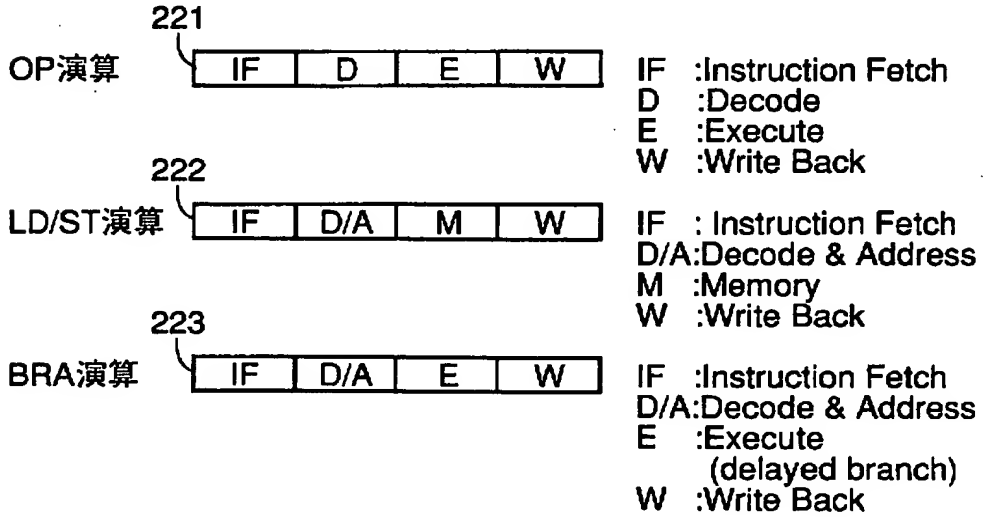


FM	サブ命令の発行数	発行順序 L-コンテナ	発行順序 R-コンテナ
00	two	1st	1st
01	two	1st	2nd
10	two	2nd	1st
11	one	1st	----

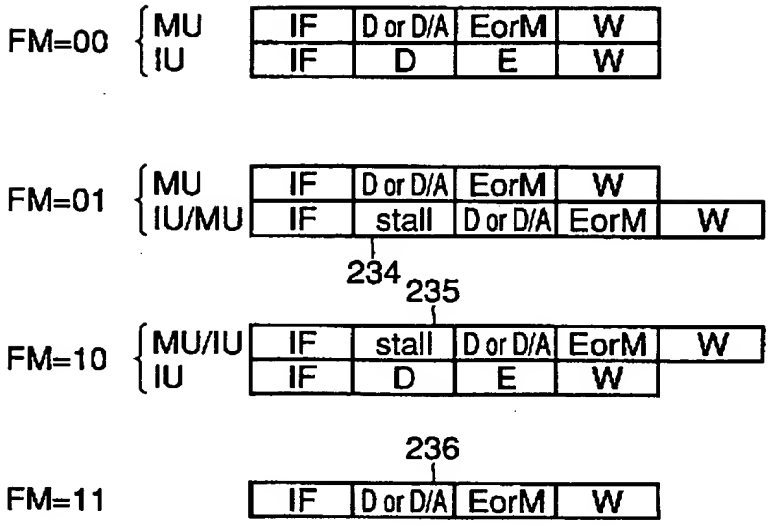
【図９】


X=00= > src= Rc
X=01= > src= Rc ; Rb++
X=11= > src= Rc ; Rb--
X=10= > src= imm:6
Y=0= > src= Rc
Y=1= > src= imm:6
F=0= > src= Rc
F=1= > src= imm:12
Z=0= > test for zero
Z=1= > test for not zero
opcode
src
Rb
Ra
X
Y
Rc
disp:18
imm:32
F
Z
ct:6
Short_M
Short_A
Short_B1
Short_B2
Short_B3
Short_D1
Short_D2
Long
211
212
213
214
215
216
217
218

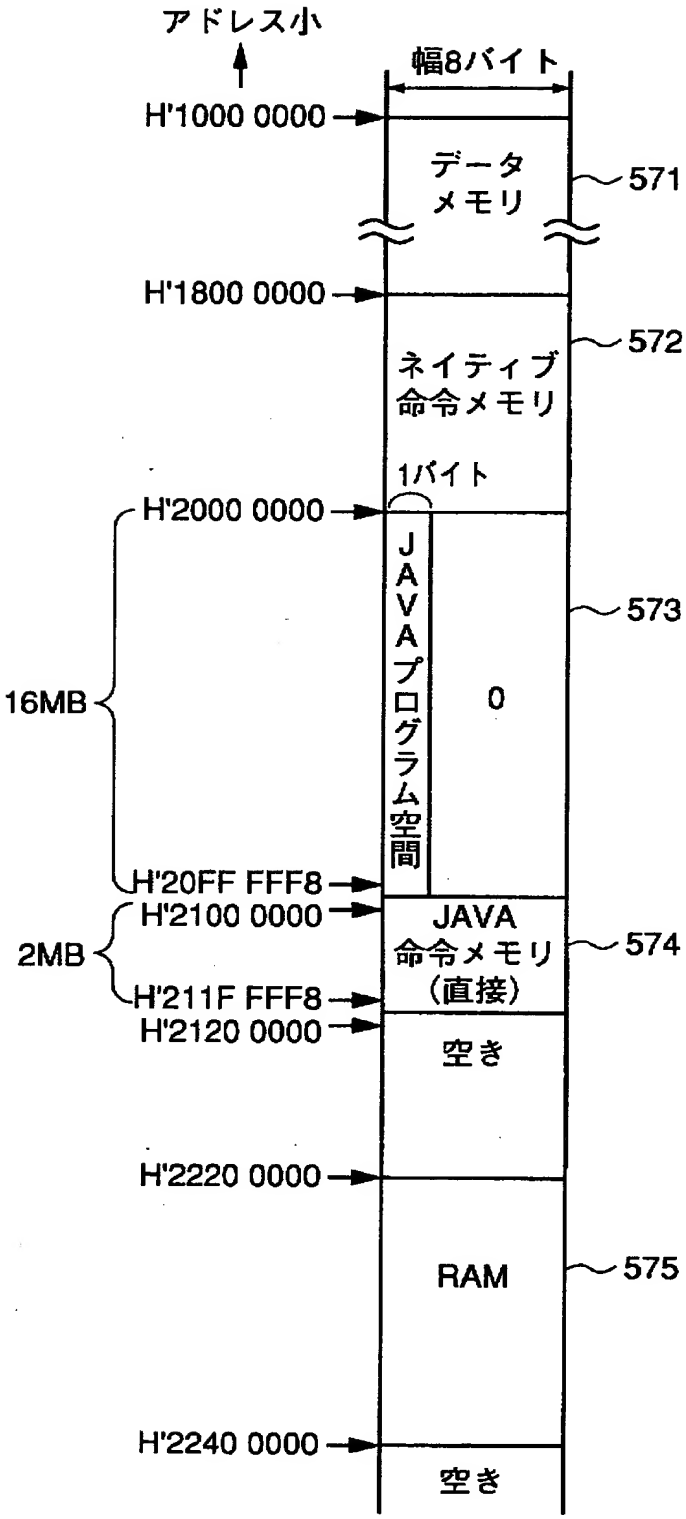
【図１０】


221
OP演算
IF
D
E
W
IF :Instruction Fetch
D :Decode
E :Execute
W :Write Back
222
LD/ST演算
IF
D/A
M
W
IF : Instruction Fetch
D/A:Decode & Address
M :Memory
W :Write Back
223
BRA演算
IF
D/A
E
W
IF :Instruction Fetch
D/A:Decode & Address
E :Execute
(delayed branch)
W :Write Back

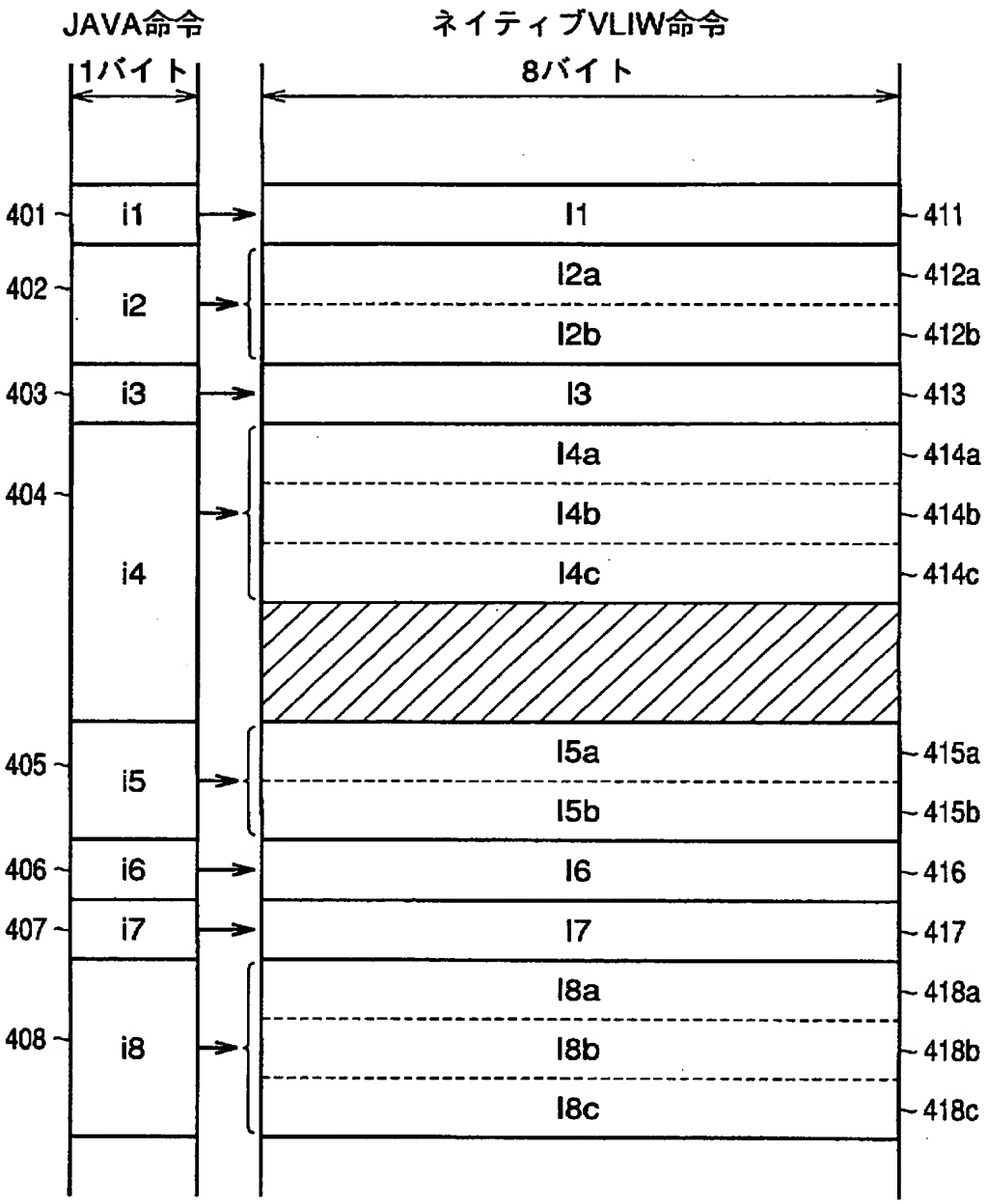
【図１１】


FM=00
MU
IU
IF
D or D/A
EorM
W
IF
D
E
W
FM=01
MU
IU/MU
IF
D or D/A
EorM
W
IF
stall
D or D/A
EorM
W
234
235
FM=10
MU/IU
IU
IF
stall
D or D/A
EorM
W
IF
D
E
W
236
FM=11
IF
D or D/A
EorM
W

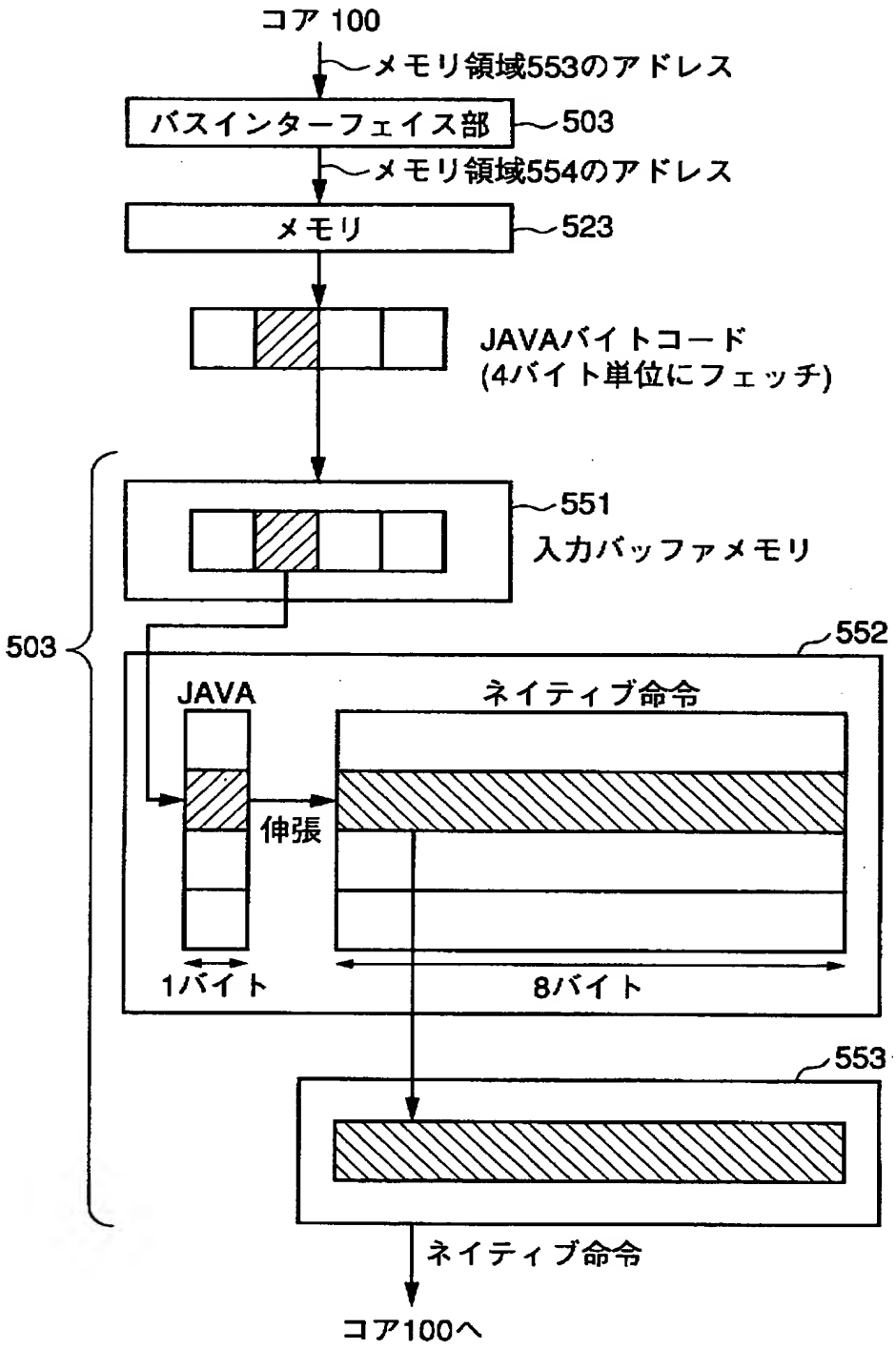
【図１２】


アドレス小
幅8バイト
H'1000 0000
データ
メモリ
571
H'1800 0000
572
ネイティブ
命令メモリ
1バイト
H'2000 0000
JAVAプログラム空間
0
573
16MB
H'20FF FFF8
H'2100 0000
JAVA
命令メモリ
(直接)
574
2MB
H'211F FFF8
H'2120 0000
空き
H'2220 0000
RAM
575
H'2240 0000
空き

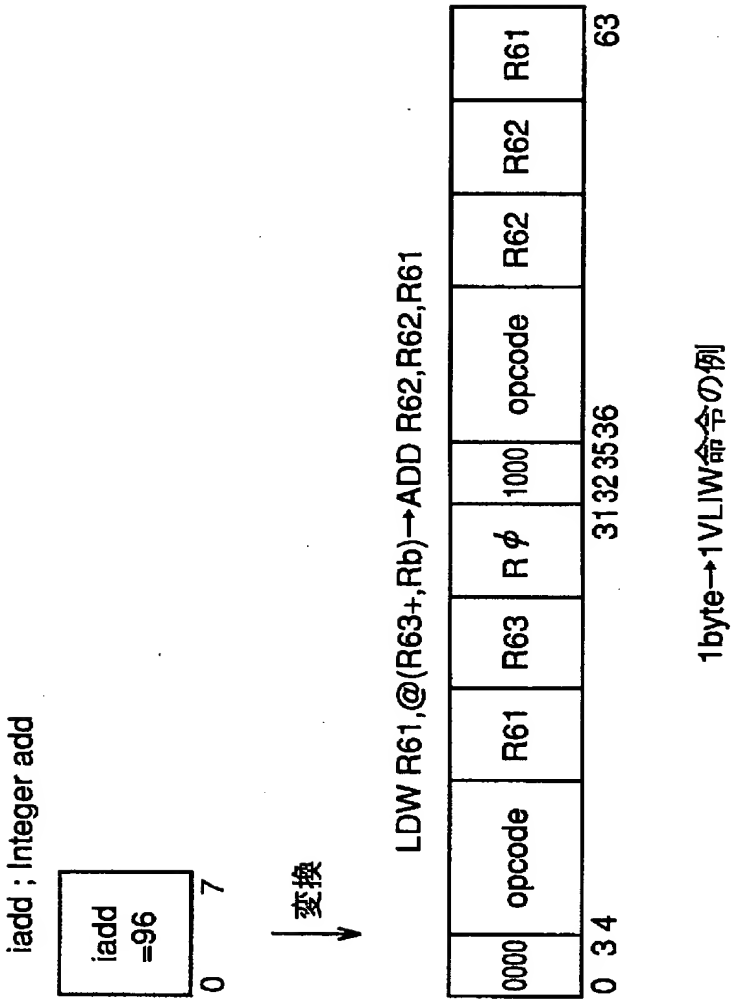
【図１３】


JAVA命令
1バイト
ネイティブVLIW命令
8バイト
401
i1
I1
411
402
i2
I2a
412a
I2b
412b
403
i3
I3
413
404
i4
I4a
414a
I4b
414b
I4c
414c
405
i5
I5a
415a
I5b
415b
406
i6
I6
416
407
i7
I7
417
408
i8
I8a
418a
I8b
418b
I8c
418c

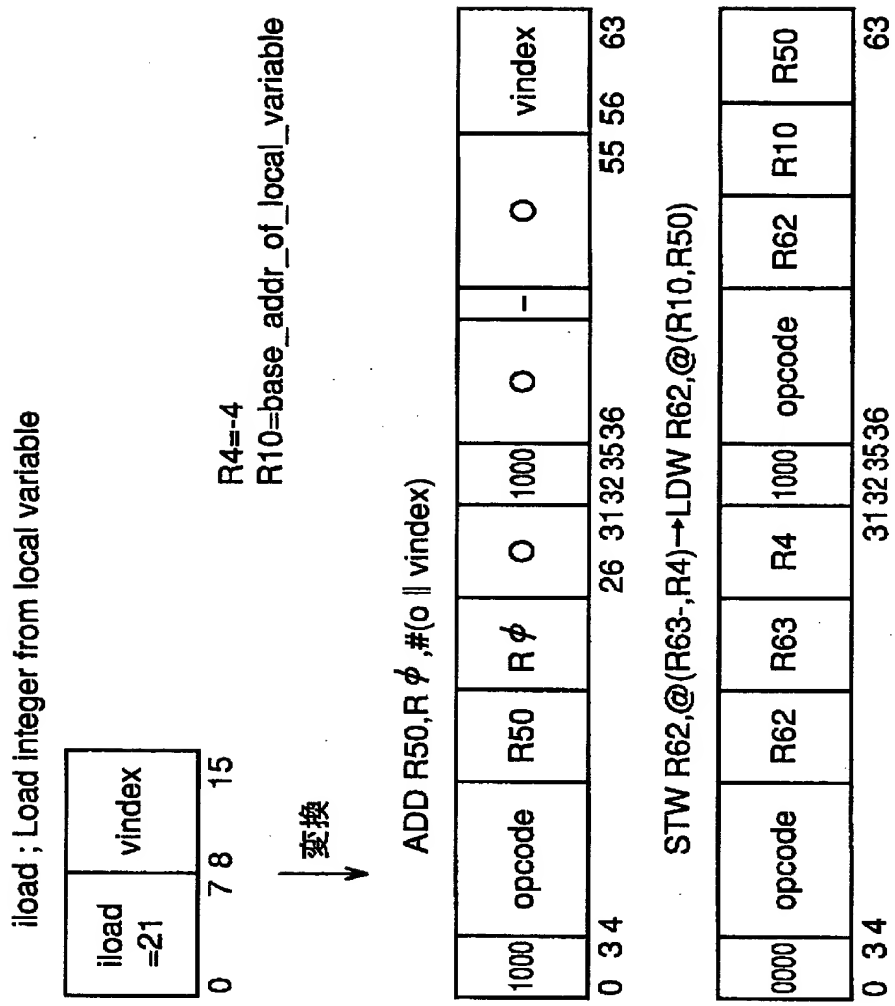
【図１４】


コア 100
メモリ領域553のアドレス
バスインターフェイス部
503
メモリ領域554のアドレス
メモリ
523
JAVAバイトコード
(4バイト単位にフェッチ)
551
入力バッファメモリ
503
552
JAVA
ネイティブ命令
伸張
1バイト
8バイト
553
ネイティブ命令
コア100へ

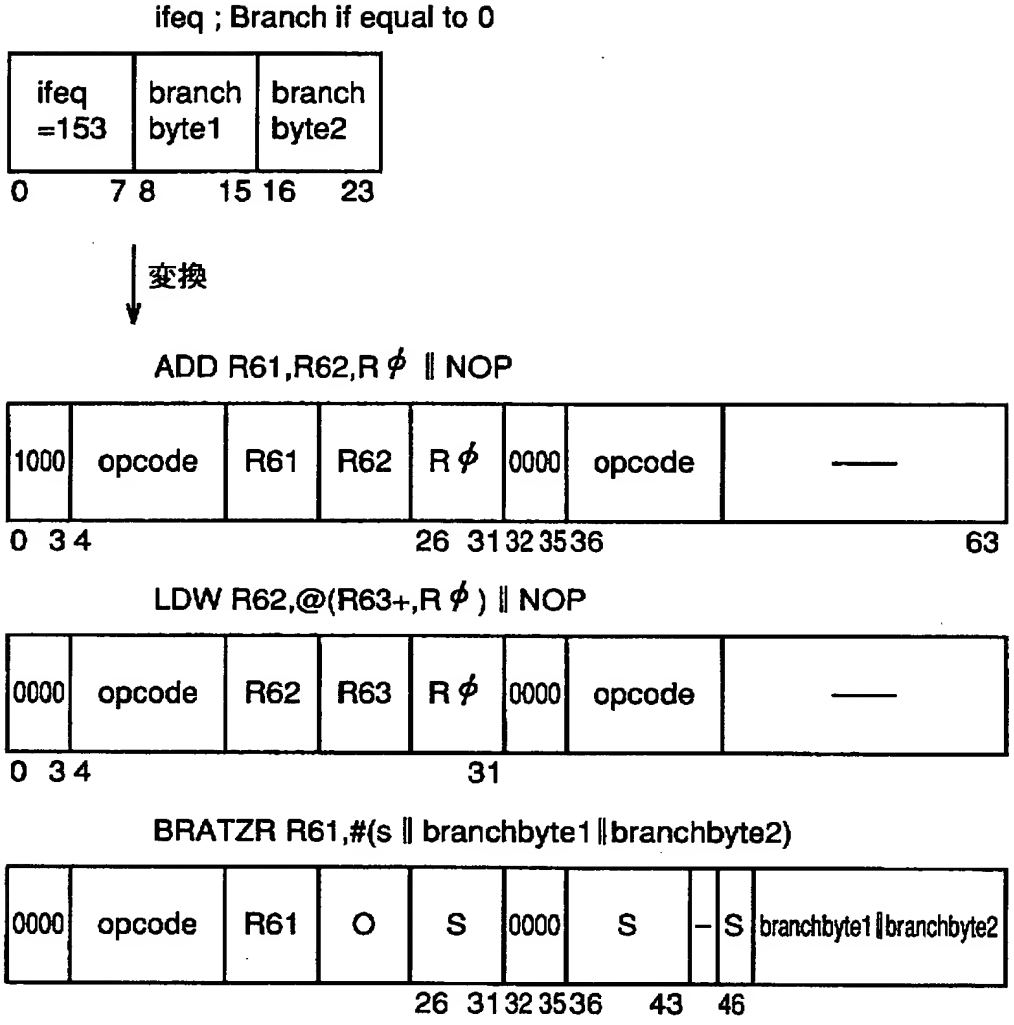
【図１５】


iadd ; Integer add
iadd
=96
0
7
変換
LDW R61,@(R63+,Rb)→ADD R62,R62,R61
0000
opcode
R61
R63
R φ
1000
opcode
R62
R62
R61
0
3
4
31
32
35
36
63
1byte→1VLIW命令の例

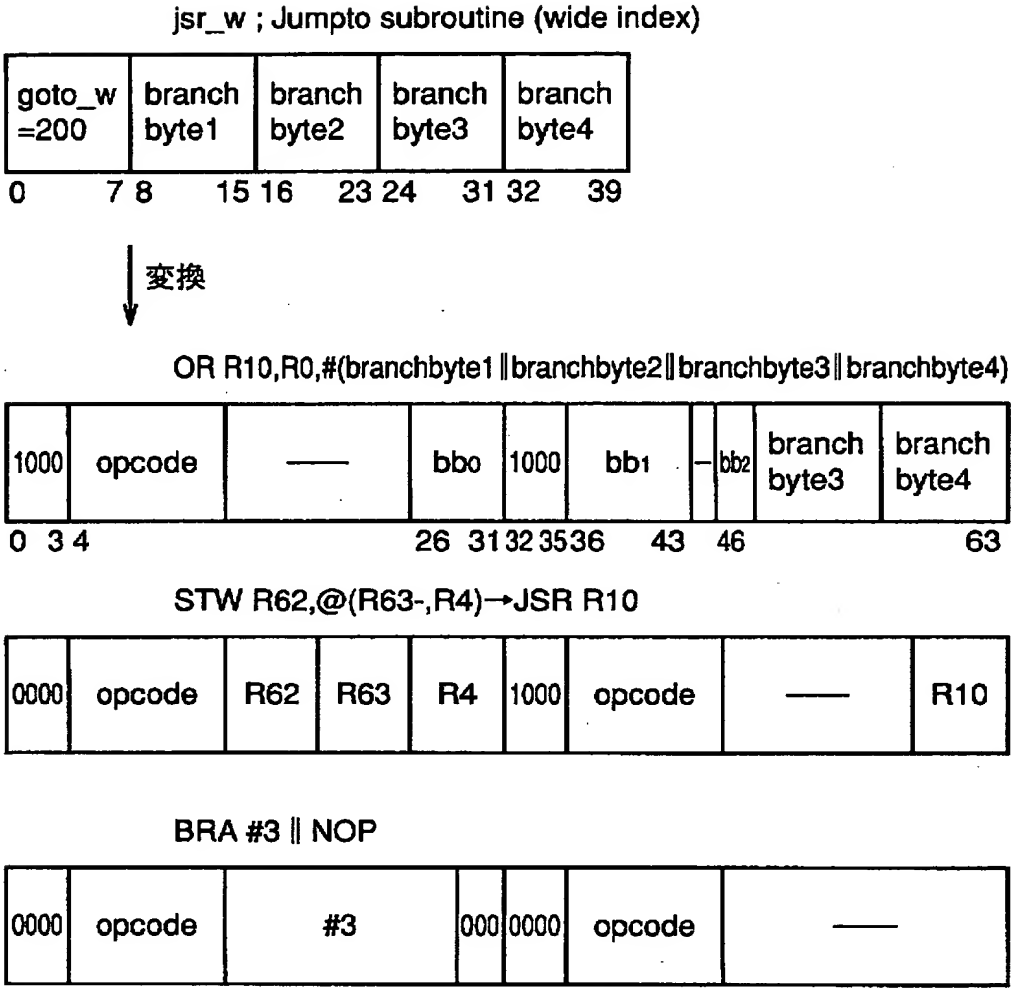
【図１６】


iload ; Load integer from local variable
iload =21
vindex
0 7 8 15
R4=-4
R10=base_addr_of_local_variable
変換
ADD R50,R ϕ ,#(o ‖ vindex)
1000
opcode
R50
R ϕ
O
1000
O
-
O
vindex
0 3 4 26 31 32 35 36 55 56 63
STW R62,@(R63-,R4)→LDW R62,@(R10,R50)
0000
opcode
R62
R63
R4
1000
opcode
R62
R10
R50
0 3 4 31 32 35 36 63
2byte→2VLIW命令の例

【図１７】

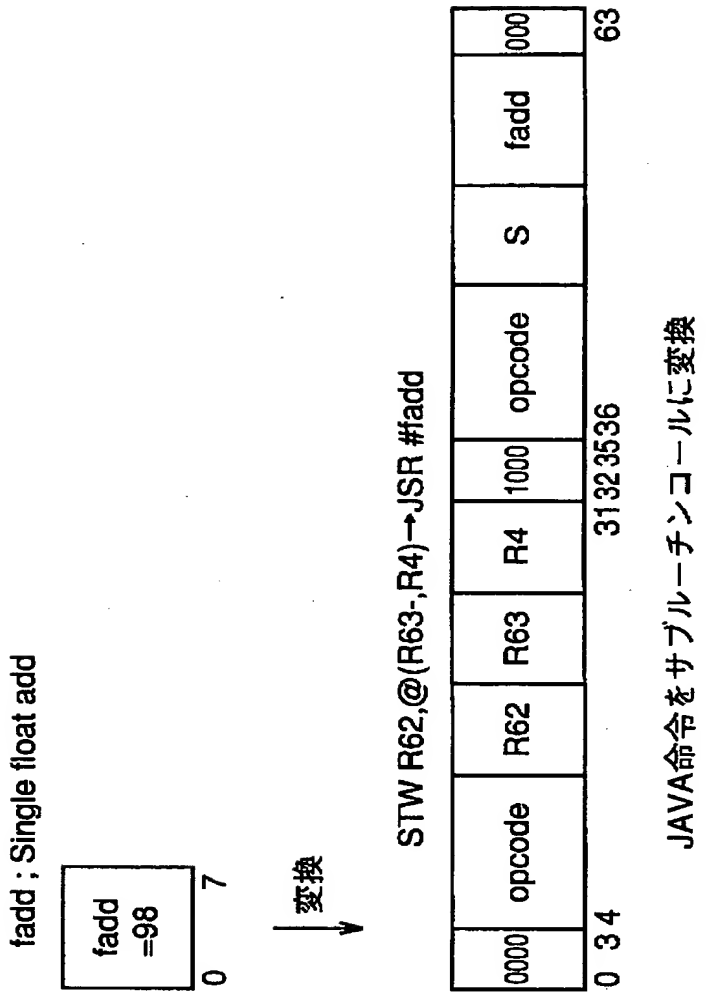

ifeq ; Branch if equal to 0
ifeq =153
branch byte1
branch byte2
0 7 8 15 16 23
変換
ADD R61,R62,Rφ ‖ NOP
1000
opcode
R61
R62
Rφ
0000
opcode
——
0 3 4 26 31 32 35 36 63
LDW R62,@(R63+,Rφ) ‖ NOP
0000
opcode
R62
R63
Rφ
0000
opcode
——
0 3 4 31
BRATZR R61,#(s ‖ branchbyte1 ‖ branchbyte2)
0000
opcode
R61
O
S
0000
S
–
S
branchbyte1 ‖ branchbyte2
26 31 32 35 36 43 46

【図１８】

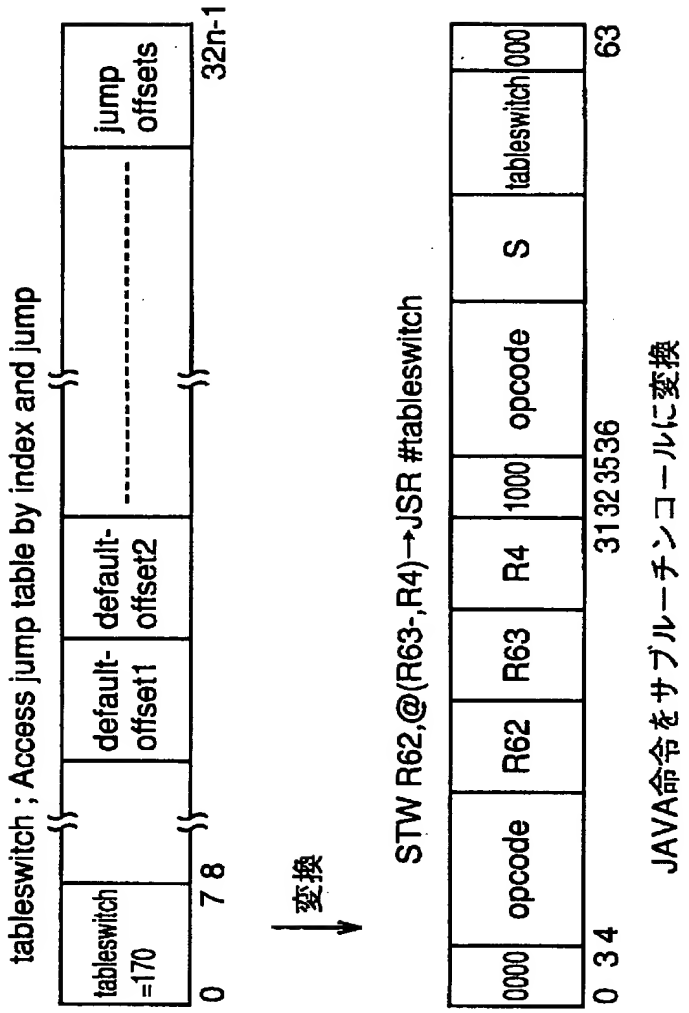


JAVA命令バイト数＞VLIW命令数

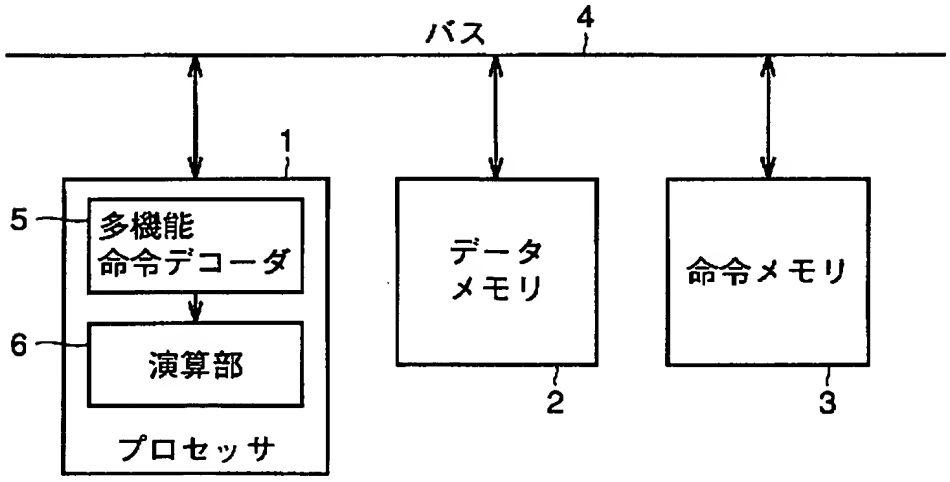
【図１９】


fadd ; Single float add
fadd
=98
0 7
変換
STW R62,@(R63-,R4)→JSR #fadd
0000
opcode
R62
R63
R4
1000
opcode
S
fadd
000
0 3 4
31 32 35 36
63
JAVA命令をサブルーチンコールに変換

【図２０】


tableswitch ; Access jump table by index and jump
tableswitch =170
default-offset1
default-offset2
jump offsets
32n-1
0
7 8
変換
STW R62,@(R63-,R4)→JSR #tableswitch
0000
opcode
R62
R63
R4
1000
opcode
S
tableswitch
000
63
31 32 35 36
0 3 4
JAVA命令をサブルーチンコールに変換

【図２１】


バス
4
1
5
多機能
命令デコーダ
6
演算部
プロセッサ
データ
メモリ
2
命令メモリ
3

出　願　人　履　歴　情　報

識別番号　　〔０００００６０１３〕

１．変更年月日　　１９９０年　８月２４日
［変更理由］　新規登録
住　所　東京都千代田区丸の内２丁目２番３号
氏　名　三菱電機株式会社

【書類名】　　要約書

【要約】

【課題】　複数個の異なる命令体系の命令からなるプログラムを高速に実行可能なデータ処理装置とそのためのメモリインターフェース装置を提供する。

【解決手段】　データ処理装置５１０のメモリインタフェース部５０３は、プロセッサコア１００から外部メモリ空間へのアクセスのためのアドレス値を受け、外部メモリから命令をフェッチするための命令フェッチ回路と、外部メモリから受ける、プロセッサコア１００に対する非ネイティブ命令をネイティブ命令に変換するための変換器と、プロセッサコア１００から外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、外部メモリ空間から読出された命令と、外部メモリ空間から読み出された命令を変換器で変換した結果の命令とのいずれかを選択的にプロセッサコア１００に与えるための選択回路とを含む。

【選択図】　　図２